

Génie Logiciel et Conduite de Projet

Plan du cours

Chapitre 1 : Rappels

Chapitre 2 : Génie Logiciel : Généralités

Chapitre 3 : Représentations du cycle de vie du logiciel

Chapitre 4 : La Démarche

Chapitre 5 : La Normalisation

Chapitre 6 : Les Tests

Chapitre 7 : La gestion des projets logiciels

Chapitre 8 : L'analyse des risques

Chapitre 9 : Conduite de changements

Chapitre 1 :Rappels

Logiciel : C'est un ensemble de programmes, procédés, règles et documentations relatifs au fonctionnement de l'ensemble de traitements d'informations.

Application : C'est un logiciel qui sert à implémenter un système d'informations.

On peut avoir 32 types d'applications :

- Applications de gestion : peu de traitement et top de donnée
- Applications scientifiques : trop de traitement et peu de donnée
- Applications industrielles : peu de traitement et peu de donnée

Système d'informations (SI): Toute organisation peut être organisée comme un système traitant des flux physiques et des flux d'information.

L'entreprise est un système ouvert sur son environnement avec lequel il échange traitant de produits, de personnel et d'argent impliquant tous les flux d'information.

A travers de la nature des flux au sein d'une entreprise, on distingue des sous systèmes assurant des fonctionnements bien spécifiques.

- Les systèmes logistiques et de production.
- Les systèmes de marketing.
- Les systèmes financier et comptable.
- Les systèmes de contrôle et de planification stratégique.

Un système d'information, est l'ensemble de flux d'opérations qu'il subisse, d'informations et de moyens mise en œuvre par se faire quelque soit la nature de ces moyens.

Le rôle d'un SI est capital par organisation d'un point de vue stratégique et opérationnel.

Un SI est le véhicule de communication d'une organisation dont il assume la formation interne et externe. Le langage de cette information est les données.

Un SI peut être constitué pour des procédures manuelles ou automatisés.

Système : c'est un ensemble d'éléments dotés d'une structure en interaction entre eux et avec leur environnement qui réalise des fonctions qui transforme la matière, l'énergie ou l'information et qui évolue dans le temps selon un objectif.

Système d'Informations Automatisé (SIA): c'est un système physique reposant sur la technologie informatique.

C'est un ensemble de logiciels élaborés à des dates différents dans des environnements informatiques qui peuvent être différents partageant certaines ressources (BD, matériel, etc..).

Méthode : "C'est un processus discipliné qui génère un ensemble de modèles décrivant les différents aspects d'un système logiciel en utilisant une certaine notation bien définie". *Grady Booch, 94*
Une méthode a pour objet de décrire l'ensemble des tâches à accomplir, l'ordonnancement des tâches, les documents et les standards qui leur sont associés à fin de prendre en charge des aspects spécifiques ou une partie du processus de développement .

Une méthode possède 4 composantes :

- Démarche
- Outil
- Model
- Langage

Un modèle est une réponse aux questions : que faire ? quand ? où ? avec qui ? et comment ?

Outil : désigne un programme ou un ensemble de programmes aidant à la mise en œuvre des techniques et donc à l'accomplissement d'une tâche. Des outils peuvent être utilisés seuls ou intégrés dans un environnement de génie logiciel.

AGL : Atelier de Génie Logiciel → CASE : Computer Aided Software Engineering
(dans CASE tout est automatisé)

Méthodologie : c'est un ensemble de méthodes et de techniques d'un domaine particulier. C'est un ensemble structuré et cohérent de méthodes, de guides et d'outils permettant de déduire la manière de résoudre un problème.

Une méthodologie désigne donc la boîte à outils du développeur intégrant les modèles de description et d'organisation, les méthodes et une organisation du projet c'est-à-dire répartition des tâches procédures de planification, estimation et conduite de projet.

Modèle : Un ensemble de concepts permettant de construire une présentation de l'entreprise.

Chapitre 2 : Génie Logiciel - Généralités

I - Définitions :

- ❖ C'est la science de l'ingénierie des logiciels. C'est la branche de l'informatique qui s'intéresse plus particulièrement à la manière dont le code source d'un logiciel est spécifié puis produit. Le complément du génie logiciel est la gestion des projets.
- ❖ « Le génie logiciel est l'application pratique de la connaissance scientifique dans la conception et l'élaboration des programmes informatiques et de la documentation associée nécessaire pour les développer, les mettre en œuvre et les maintenir » Barry Boehm, 1977
- ❖ Le génie logiciel est le domaine des sciences de l'ingénieur dont la fonctionnalité est la conception, la fabrication et la maintenance des systèmes logiciels complexes, sûrs et de qualité. Ces systèmes sont caractérisés par un ensemble de documents de conception de programmes et de jeux de test avec souvent de multiples versions.

Remarque : Le génie logiciel touche du cycle de vie du logiciel donc concerne toute les phases de la création d'un logiciel informatique

1. Etude et analyse du besoin
2. Elaboration des spécifications
3. Conceptualisation du mécanisme interne au logiciel
4. Conceptualisation de techniques de programmation
5. développement (programmation)
6. test
7. mise en oeuvre et maintenance

II - Objectifs du GL :

Le GL vise à utiliser l'informatique dans les entreprises comme outils de tous les jours, il vise à utiliser le développement et la maintenance c'est-à-dire réduire les coûts et assurer la qualité grâce à l'utilisation des méthodes, d'outils et de méthodologies. Ces objectifs peuvent être atteints que si certains principes sont respectés.

III - Principes du GL :

- ❖ **Rigueur** : Grande exactitude dans l'application des règles.
- ❖ **Séparation des problèmes** : Il s'agit d'une séparation dans le temps, d'une séparation de vues* dans le système et d'une séparation en sous-systèmes.
*Exemple Orienté Objet : Dynamique/statique, Merise : Données/traitements
- ❖ **Modularité** : L'évolution des langages de programmation, en particulier les langages OO, visent à rendre plus facile une programmation modulaire appelée programmation par composants.
- ❖ **Abstraction** : on nous considère à un moment donné que les objets jugés importants du système. Une même réalité peut être décrite à différents niveaux d'abstraction.
- ❖ **Généricité** : c'est le fait pour un objet de pouvoir être utilisé tel qu'il est, dans différents contextes ou même indépendamment du contexte.
Exemple on peut citer le comportement d'une pile sans savoir le type de données contenues.
- ❖ **Construction Incrémentale** : c'est le fait de réaliser l'objectif par des étapes successives.
Exemple Réalisation d'un noyau de fonctions essentielles, puis ajout des fonctions secondaires.
- ❖ **Anticipation du changement** : Prévoir et faciliter les changements en favorisant la construction modulaire et en gérant les configurations (traces de ce qu'on a modifié) des modèles.

Remarque : Chaque méthodologie privilégie certains principes et le choix d'une méthodologie pour un projet va dépendre des objectifs visés. **Aucune méthodologie ne va couvrir tous les principes qui doivent être suivis.**

IV - Les attributs du logiciel :

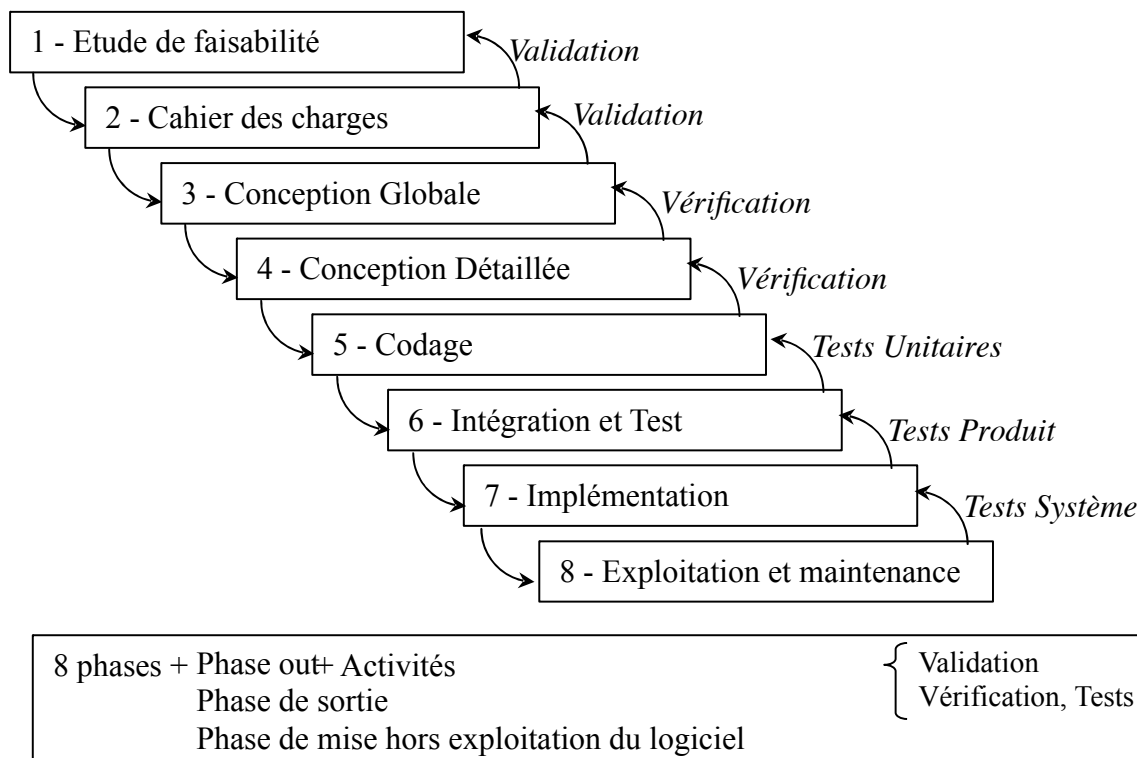
- ❖ **Le couplage :** Mesure des indépendances entre modules : plus que le couplage est fort, plus que les modules sont dépendants. (Couplage fort → contre la modularité)
- ❖ **La complexité :** le degré de complexité d'un système. Plus qu'on a des interconnexions entre modèles (degré d'intégrité élevé), plus que le programme est complexe.
- ❖ **L'extensibilité :** le degré d'accommodation aux changements produits par une nouvelle exigence à réaliser.
- ❖ **La lisibilité :** Mesure la difficulté de comprendre un composant logiciel. Cette lisibilité est liée à la complexité et à la documentation.
- ❖ **Bonne définition des interfaces :** Mesure le degré de dépendance fonctionnel entre les tâches réalisées par un module (Interface Utilisateur, Interface Inter Logiciels, Interface Inter Modules u sein d'un même logiciel).
- ❖ **Degré d'abstraction des classes :** Une classe est caractérisée par les opérations qu'elle permet d'effectuer.
- ❖ **Visibilité du comportement :** C'est la compréhension des composants du logiciel. C'est une mesure qui représente la possibilité de se rendre compte de façon aisée de ce font les différentes fonctionnalités d'un programme.

Chapitre 3 : Les Représentations du cycle de vie du logiciel

Le développement d'un logiciel n'est pas simplement l'écriture d'un programme sur une durée variant de heures à quelques jours, c'est un processus beaucoup plus complexe auquel on applique la règle de DECART : « Diviser pour régner ».

Le cycle de vie d'un logiciel débute par l'analyse du problème, comprend le processus complet de développement (création), et le processus complet de maintenance. Un cycle de vie se décompose en deux sous cycles : le cycle de développement et le cycle de maintenance.

I - Le modèle en cascade (en chute d'eau) :



❖ **Validation** : C'est le fait d'établir l'utilité, c'est la réponse à la question : construisons nous le produit qu'il faut ?

❖ **Vérification** : C'est le fait d'établir la cohérence, c'est la réponse à la question : construisons nous le produit comme il faut ?

Remarque : Ce modèle suppose que le cahier des charges ne peut plus être modifié d'une façon substantielle après sa validation.

Remarque 2: Dans certains projets on ne peut pas spécifier les besoins dès le départ.

1) Etude de faisabilité

Objectifs :

- Comprendre le problème
- Etudier la faisabilité technique et économique
- Analyser les besoins

Résultats :

- Décision sur la faisabilité
- Estimation partielle des coûts et des délais
- Plan global u projet
- Cahier des charges préélémentaire

2) Cahier des charges

Cette phase débouche sur un cahier des charges. Le cahier des charges en plus de son rôle comme un descripteur des besoins du client, sert de documents contractuelles et juridiques entre le client et développeur.

Le prototype : c'est la solution idéale pour un cahier des charges précis et le moins incomplet possible, c'est le développement d'un sous problème complémentaire au problème global à traiter et à examiner l'appréciation du client vis-à-vis au prototype afin de préciser au mieux ses besoins sans omission et sans obligation. Le prototype réalise partiellement des fonctions sont totalement l'interface utilisateur.

3) Conception Globale

Objectifs : - Décrire l'architecture du logiciel

- Obtenir une description du système comme un ensemble de modules et de structures de données.
- Rôle de chaque module et interface modulaire

Résultats : - Structure modulaire du système
- Structure de données

4) Conception Détaillée

Algorithme + Pseudo Code

Objectif : - Obtenir une description détaillée des traitements et des structures de données par des expressions traduisibles directement dans un langage de programmation choisi.

Résultats : - Structure modulaire détaillée
- Interface précise bien explicitée
- Algorithme de chaque module avec structure de donnée utilisée

5) Codage

Objectif : - Obtenir le programme et faire les tests unitaires.

Moyen : - Langage de programmation.

Résultats : - Programme
- Documents techniques
- Résultat des tests unitaires

6) Intégration et Test

Cette phase consiste à faire coopérer et à assembler les composants modulaires testés individuellement. L'intégration se fait graduellement, il s'agit d'une intégration incrémentale. La difficulté est proportionnelle au nombre de modules.

Résultats : - Programme
- Documents techniques
- Résultat des tests unitaires

7) Implémentation ou installation ou implantation

Objectif : - Mise en exploitation et l'utilisation du logiciel dans l'environnement réel

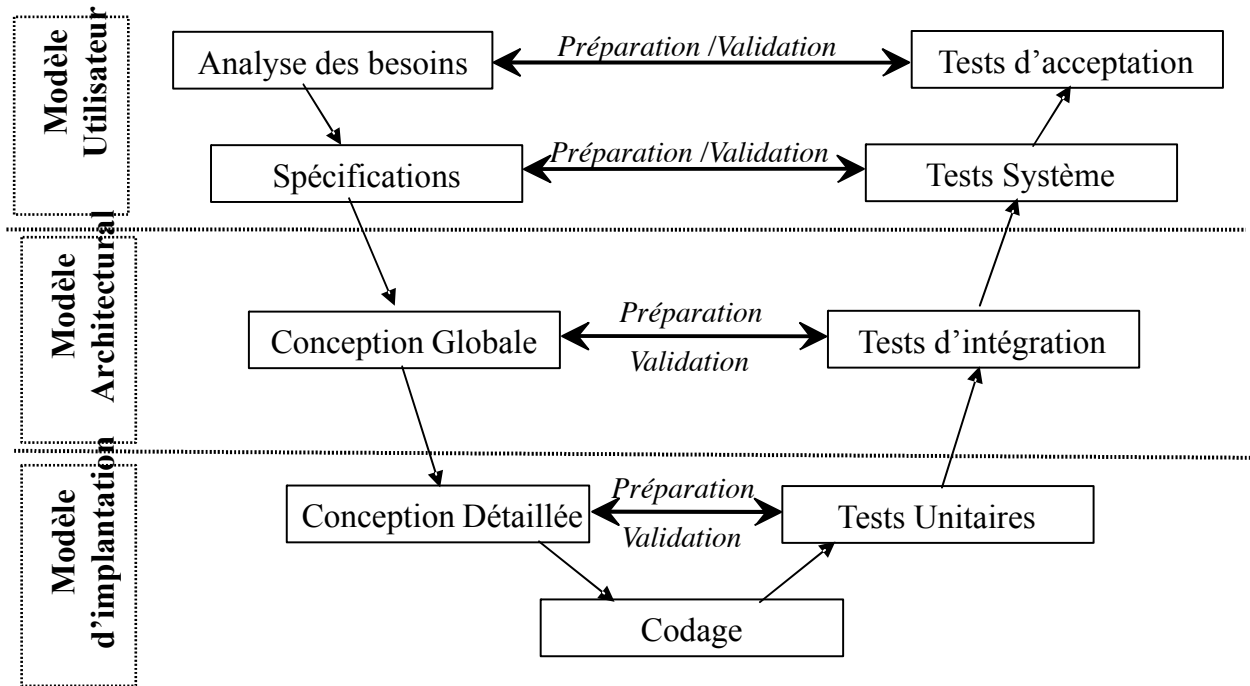
Résultats : - Logiciel installé, testé et accepté par le client
- Manuel d'utilisation fourni
- Formation aux utilisateurs planifiés

8) Maintenance

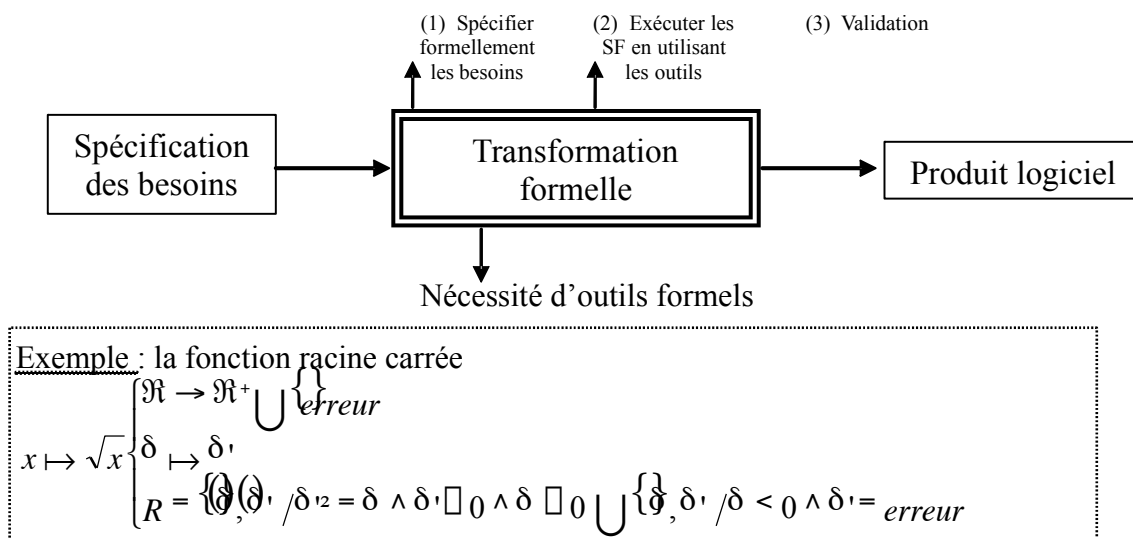
On peut classer la maintenance en 5 catégories :

- **La maintenance corrective :** Qui traite les erreurs et les défaillances du programme.
- **La maintenance adaptative :** Qui traite les adaptations suite aux changements dans l'environnement.
- **La maintenance perfective :** Qui traite les améliorations de performance.
- **La maintenance évolutive :** Qui consiste à modifier les missions spécifiées du logiciel.
- **La maintenance préventive :** Qui consiste à enrichir le code avec des mécanismes de traitement d'exceptions.

II - Le modèle en V :



III - Le modèle de la transformation formelle :

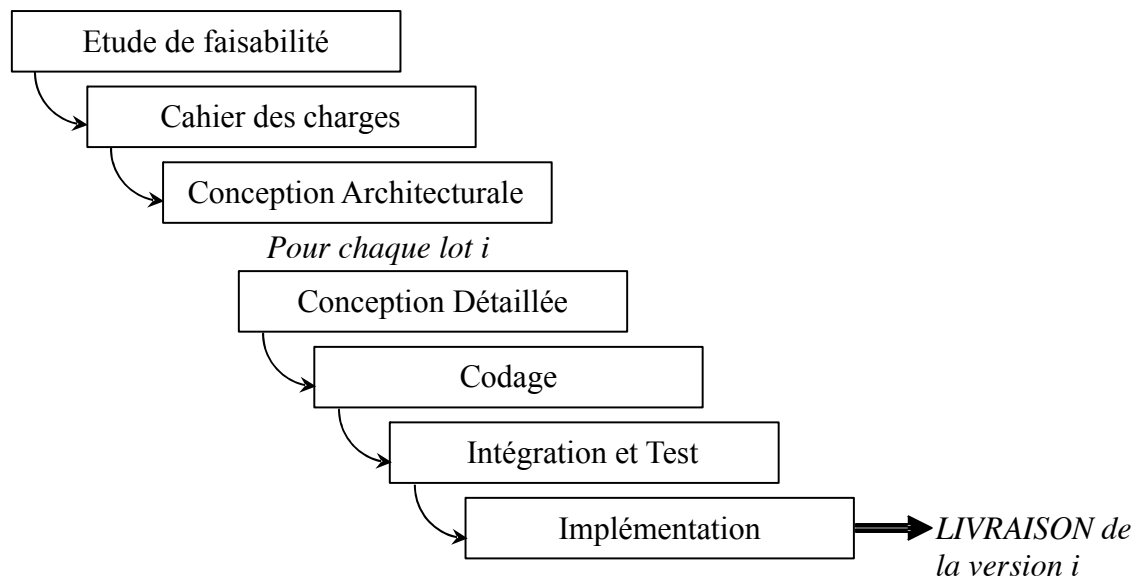


IV - Le modèle incrémental :

Consiste à développer les futurs logiciels par lot et établir un planning de réalisation de ces lots avant le démarrage du parcours de construction.

Chaque lot doit fournir un produit qui sera immédiatement mis en exploitation dans un environnement opérationnel.

L'ensemble des produits en exploitation à une période donnée constitue une version provisoire du logiciel dont la durée de vie s'étend jusqu'à la livraison du prochain lot.



V - Le modèle de prototypage :

Définition : prototype

- *Le robot* : c'est un type, un modèle premier (original ou principal). Un premier exemplaire d'un modèle construit avant la fabrication en série
- C'est un modèle réduit, un artéfact du logiciel de faible coût, développé pour évaluer certains aspects d'un système informatique, c'est modèle de système à réaliser où l'on sacrifie à la précision afin de permettre une vérification rapide du comportement de ce logiciel.
- Un prototype est un modèle exécutable de tout ou d'une partie d'un logiciel facile à mettre en œuvre et à modifier qui va permettre de vérifier rapidement certaines fonctionnalités de ce logiciel en sacrifiant la précision des autres. Il pourra ne pas être construit avec les mêmes contraintes de robustesse que le système final.

Un prototypage est une approche de développement du logiciel privilégiant la préparation de version de travail utilisée comme base d'évaluation d'idées, de décisions en vue de préparer la version complète livrable au client.

Le prototypa c'est :

- L'élaboration de versions opérationnelles du futur logiciel dès le début du cycle de vie
- La mise en évidence et la clarification par expérimentation des problèmes les plus significatifs
- L'élaboration de prototype qui constitue une base de discussion et de communication entre les utilisateurs, les informaticiens et les autres acteurs de l'organisation.

Un prototypage est une approche de développement qui peut être associée avec d'autres modèles de développement et qui permet de valider les besoins d'utilisateurs, les choix techniques, l'ergonomie de l'interface utilisateur, la faisabilité ou le comportement du futur logiciel à partir de petites applications significantes (prototype).

Les phases du cycle de vie d'un prototype sont les suivantes :

- Analyse et spécification des besoins.
- Conception (pas robuste) et réalisation
- Evaluation du prototype
- Révision et modification du prototype
- Compléter les définitions des besoins

Avantages

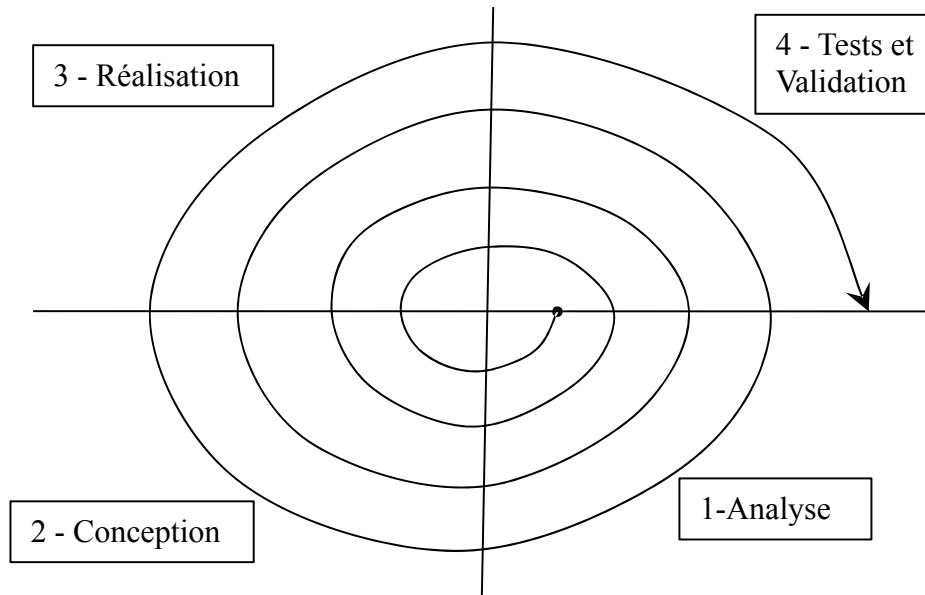
- Permet de se concentrer sur les points critiques et les zones d'incertitude très tôt dans le développement.
- Simplifie l'élaboration des spécifications et de l'interface Homme/Machine

- Evolutif et incrémental
- Il peut être utilisé avec d'autres modèles et à toutes les phases, selon la phase, les objectifs du prototype peuvent être différents

Inconvénients

Beaucoup de problèmes liés à la gestion de projet (coût, temps, suivie, contrôle, planification ...)

VI - Le modèle en spirale:



Ce modèle a été introduit en 1980, il est non linéaire et anti-risques. Il se déroule selon une suite de développement de 4 phases dont le nombre n'est pas déterminé à l'avance et dépend de l'approche de développement choisie. Le choix de l'approche de développement, des méthodes et des techniques mis en œuvre est déterminé en fonction des risques analysés et identifiés au début de chaque cycle du spiral.

- Au cours de la phase de l'analyse, les alternatives et les contraintes.
- Au niveau de la conception, on analyse les risques et on évalue les alternatives.
- Au niveau de la réalisation, on prépare un prototype
- Au niveau des tests et des évaluations, on valide ce qui était réalisé au cours du cycle de spiral et on planifie le prochain cycle.

Le modèle de la spirale est un modèle mixte ou méta modèle, il permet de mettre en œuvre des approches, des méthodes et des techniques variées (prototypage, développement linéaire ...) au cours des différentes phases d'un cycle de la spirale.

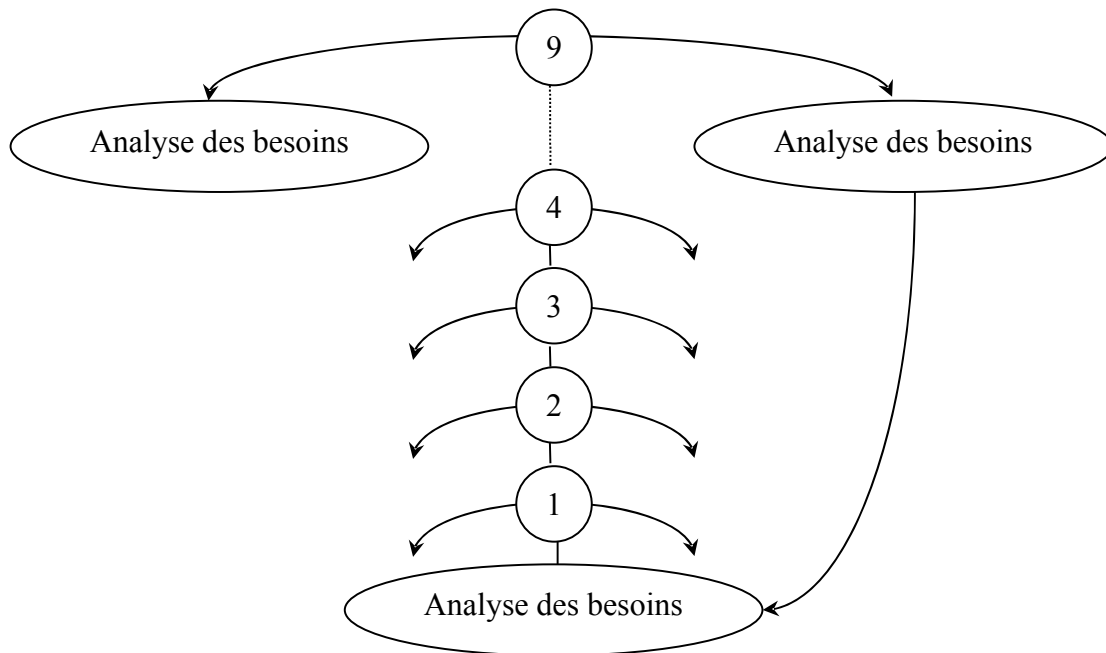
Remarque : Le modèle en spirale n'est pas fondamentalement différent du modèle évolutif.

VII - Le modèle de la fontaine :

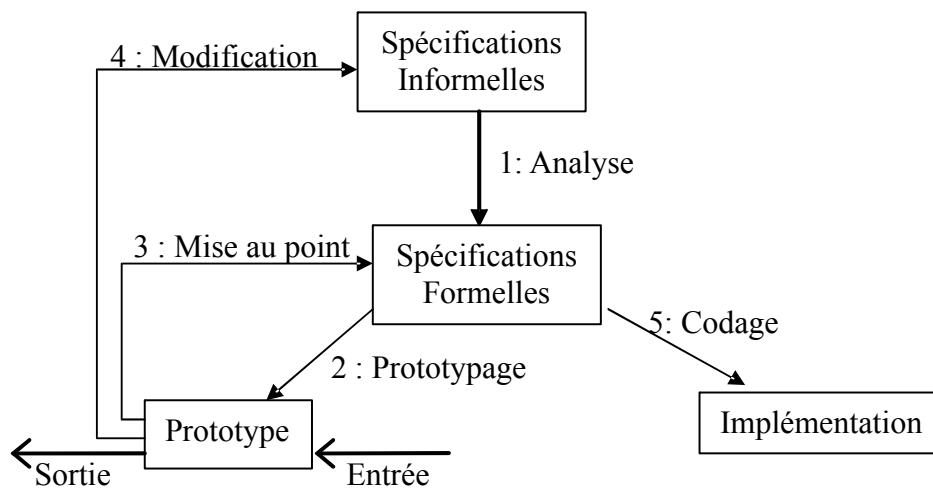
Ce modèle est orienté objet, il est incrémental et itératif.

<i>Phases Analytiques</i>	<i>Phases Synthétiques</i>
1- Spécification des exigences du <u>système</u>	4- Conception Architecturale
2- L'indentification des objets	5- Conception des classes
3- Identification des interactions entre objets	6- Identification des généralisations / Spécifications
	7- Codage
	8- Test du logiciel
	9- Utilisation
	10- Maintenance / Développements ultérieurs

Remarque : Chaque étape peut fournir des résultats modifiant les résultats suivants.



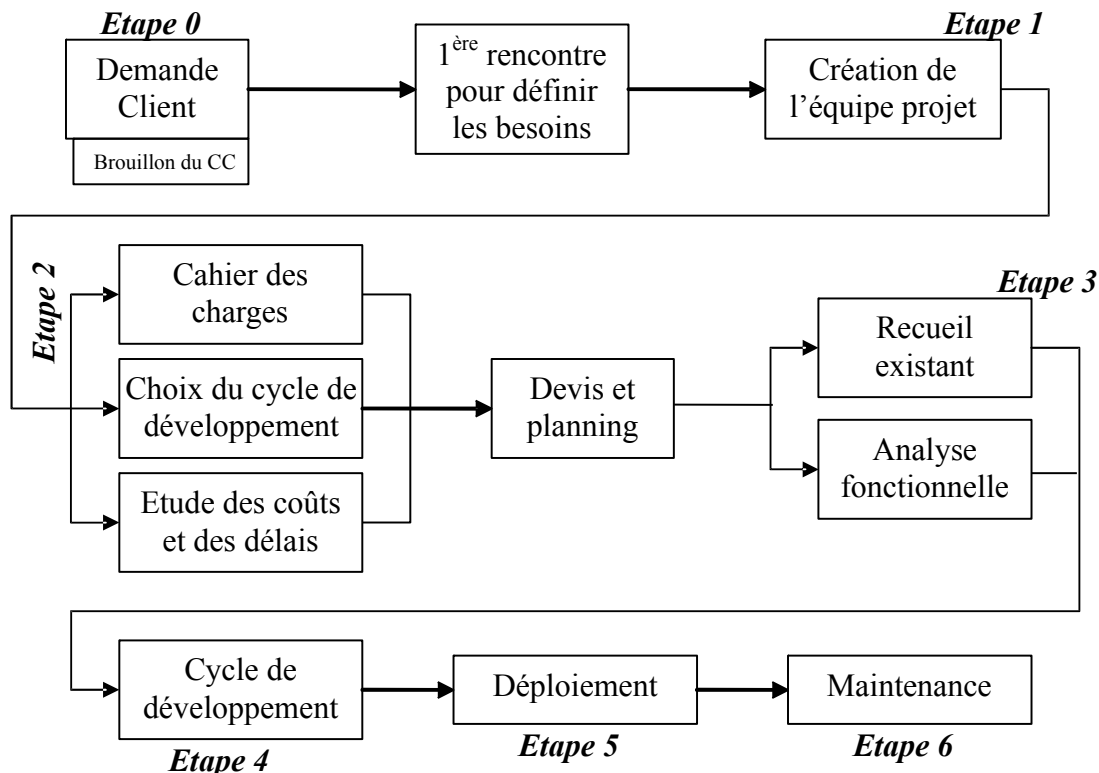
VIII - Le modèle de Balzer :



Avec les méthodes formelles, les cycles de vie classiques ne marchent plus, il faut les adapter. Le cycle de vie de Balzer représente la nouvelle famille de cycles de vie adoptés au développement formel.

Chapitre 4 : La Démarche

I - Présentation d'une démarche classique :



II - Rappels : Les méthodes de développement semi formelles

- **SADT (Structure Analysis and Design Technique) :** C'est une méthode d'origines américaine utilisée à grande échelle vers la fin des années 80 et considérée comme standard de description graphique d'un système complexe par analyse fonctionnelle descendante.
- **MERISE :** c'est une méthode d'origine française apparue au années 70, c'est une approche systémique avec séparation entre données et traitements et elle représente 4 niveaux d'abstraction.
- **Processus Unifié et UML :** c'est une méthode « Itérative et Incrémentale », Orientée Objet. Cette méthode a deux approches (Ascendante et Descendante) et intègre 4 phases et des activités dans chaque phase.
- **OMT (Object Modelling Technique) :** Ces techniques sont l'œuvre de James Rumbaugh qui est l'un des pères d'UML. Les concepts de base de ces techniques sont : les classes, les instances, les attributs et les méthodes...
- **OOSE :** c'est l'œuvre de Ivar Jacobson, elle est basée sur 5 modèles :
 - Le modèle des besoins
 - Le modèle d'analyse
 - Le modèle de conception
 - Le modèle d'implémentation
 - Le modèle de Test

Remarque : OMT et OOSE étaient à l'origine d'UML.

- **JSD (Jackson Software Development) :** c'est une technique de développement basée sur une spécification fonctionnelle et sur l'implémentation finale
- **OOA (Oriented Object Analysis) :** c'est une méthode d'analyse Orientée Objet utilisée surtout dans l'industrie et la gestion.
- **HOOD :** c'est une méthode de conception orientée objet qui met l'accent sur les points de vue statique et dynamique.

III - Les méthodes de développement formelles

On a besoin des méthodes les plus rigoureuses pour certains domaines :

- Sécurité et certification
- Systèmes embarqués (voitures, véhicules...)
- Automates (domaine médical)

Les méthodes formelles garantissent la correction du logiciel, élimine les erreurs et les dysfonctionnements et facilitent la maintenance. Dans les méthodes formelles on trouve la logique, l'algèbre, la mathématique discrète, la théorie des ensembles, la théorie des automates...

Exemple d'application industrielle de la méthode B

- Système de contrôle de vitesse du train de SNCF
- Métro de Monreale et de Marseille
- Métro sans conducteur
- ...

Une méthode formelle intègre un langage formel et un système de preuve ou de raisonnement formel.

IV - Le langage Z de spécification formelle

1) Définition

De point de vue sémantique, le langage Z est subdivisé en 3 parties :

1. Un langage mathématique utilisant les propositions la logique des prédicats, les ensembles et les relations...
2. Un langage schématique représentant les manipulations algorithmiques des données ou des objets.
3. Une théorie de raffinement entre les types abstraits de données.

Z a été développé à l'université d'Oxford à la suite des travaux de Jean Rene Abrial.

On parle souvent de Z comme un langage de description d'état en utilisant un dialecte mathématique particulier dans une notation schématique, il décrit les différents états de données par lequel peut passer un produit logiciel et sous quelles conditions.

2) Rappels

Proposition : c'est une expression qui est vraie ou fausse.

Les *connecteurs* par ordre de priorité sont : Non, ET, OU, Implique et équivalent

Prédicat : expression contenant des inconnues qui se transforme en propositions quand celles-ci sont identifiés.

Déclaration

$x : a \rightarrow$ on déclare que la variable x appartient à l'ensemble a

Qualification

$Q x : a \bullet p \rightarrow$ soit p un prédicat impliquant la variable x élément de a à travers cette écriture, on retransforme le prédicat en proposition en appliquant le quantificateur Q à x .

$\forall x : a \bullet p \Rightarrow p$ est une proposition vraie pour tous les éléments de a

$\exists x : a \bullet p \Rightarrow p$ est une proposition vraie pour certains éléments de a

$\exists_1 x : a \bullet p \Rightarrow p$ est une proposition vraie pour un seul éléments de a

Contraintes

Il est possible d'ajouter un prédicat δ à la déclaration d'une proposition quantifiée pour restreindre le domaine d'une variable.

$Q x : a \mid r \bullet p \rightarrow p$ une proposition vraie si $x : a$ vérifie la contrainte r .

Notation μ

Soit une déclaration D , et un prédicat P exprimant une contrainte sur les variables déclarés en D et E une expression donnant facultativement la valeur qui on désire obtenir.

On écrira la description définie $\mu D \mid P \bullet E$.

Exemple : le cube de l'entier > 0 dont le carré = 25 s'écrit de cette manière $\mu x = 2 \mid x > 0 \wedge x^2 = 25 \bullet x^3$

Ensemble en Z

On peut bâtir les ensembles de plusieurs manières :

- Par extension : on se contente de faire la liste des éléments. *Exemple* {0, 1, 4, 9, 16}

- Par compréhension : on pourra définir un ensemble sous forme $\{D|P \bullet E\}$. *Exemple* $\{x:IN \mid x \leq 4 \bullet x^2\}$

Opération

Les opérations sur les ensembles en \mathbf{Z} sont :

: cardinal exemple : # {1,2,3,4} = 4

U : union

\cap : intersection

\setminus : différence exemple : $\{1,3,6\} \setminus \{2,4,6\} = \{1,3\}$

Abréviation

Un objet e déjà défini dans une spécification formelle peut se voir attribuer le nom x par abréviation.

$x == e$

exemple : $\{x:IN \mid x \leq 5 \bullet x^2\}$ a sera donc l'ensemble {1,2,4,9,16}

Autres notations particulières liées à IN et Z

succ x : donne le successeur de x

a..b : l'ensemble de tous les valeurs comprises entre a et b au sens large. exemple : $3..5 = \{3,4,5\}$

max et **min** : donnent respectivement la plus grande valeur et la plus petite valeur d'un ensemble.

exemple : $\max \{x:IN \mid x < 5 \bullet x^2\} = 16$

3) Les relations

X, Y, R
 $\mathfrak{R} : IP(X \times Y)$

$X \leftrightarrow Y == IP(X \times Y)$
 $\mathfrak{R} : X \leftrightarrow Y$

* **dom** : définit le domaine d'une relation

$\mathfrak{R} : X \leftrightarrow Y$

$x \mapsto y$

exemple : $\text{dom } \mathfrak{R} == \{x : X, y : Y \mid x \mapsto y \bullet x\}$

* **ran** : définit l'image d'une relation

exemple : $\text{ran } \mathfrak{R} == \{x : X, y : Y \mid x \mapsto y \bullet y\}$

* $\mathfrak{R} \sim$: définit le domaine d'une relation

exemple : $\mathfrak{R} \sim == \{x : X, y : Y \mid x \mapsto y \bullet y \mapsto x\}$

* Restrictions de domaine et d'images

$\mathfrak{R} : X \leftrightarrow Y$

A, B

$A \triangleleft \mathfrak{R} == \{x : X, y : Y \mid x \mapsto y \in \mathfrak{R} \wedge x \in A \bullet x \mapsto y\}$

$B \triangleright \mathfrak{R} == \{x : X, y : Y \mid x \mapsto y \in \mathfrak{R} \wedge y \in B \bullet x \mapsto y\}$

* soustraction de domaine et d'images

$\mathfrak{R} : X \leftrightarrow Y$

A, B

$A \triangleleft \mathfrak{R} == \{x : X, y : Y \mid x \mapsto y \in \mathfrak{R} \wedge x \notin A \bullet x \mapsto y\}$

$B \triangleright \mathfrak{R} == \{x : X, y : Y \mid x \mapsto y \in \mathfrak{R} \wedge y \notin B \bullet x \mapsto y\}$

* image relationnelle

$\mathfrak{R}(|A|) == \text{ran}(A \triangleleft \mathfrak{R})$

$\mathfrak{R} : X \leftrightarrow Y$

Exemple :

$\mathfrak{R} : X \leftrightarrow Y$

$\mathfrak{R} == \{x : X, y : Y \mid x < 6 \wedge y = x^3 \bullet x \mapsto y\}$

$= \{x : \mathbb{N} \mid x < 6 \bullet x \mapsto x^3\}$

$= \{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 8, 3 \mapsto 27, 4 \mapsto 64, 5 \mapsto 125\}$

$$\text{dom } \mathfrak{R} = \{x : \mathbb{N} \mid x < 6 \bullet x\} = \{0, 1, 2, 3, 4, 5\}$$

$$\text{ran } \mathfrak{R} = \{x : \mathbb{N} \mid x < 6 \bullet x^3\} = \{0, 1, 8, 27, 64, 125\}$$

$$\mathfrak{R} \sim = \{x : \mathbb{N} \mid x < 6 \bullet x^3 \mapsto x\} = \{0 \mapsto 0, 1 \mapsto 1, 8 \mapsto 2, 27 \mapsto 3, 64 \mapsto 4, 125 \mapsto 5\}$$

$$A = \{x : \mathbb{N} \mid x < 4 \bullet x\} = \{0, 1, 2, 3\}$$

$$B = \{x : \mathbb{N} \mid x^3 < 25 \bullet x\} = \{0, 1, 8\}$$

$$A \triangleleft \mathfrak{R} = \{x : \mathbb{N} \mid x < 4 \wedge x \mapsto x^3 \in R \bullet x \mapsto x^3\} = \{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 8, 3 \mapsto 27\}$$

$$B \triangleright \mathfrak{R} = \{x : \mathbb{N} \mid x^3 < 25 \wedge x \mapsto x^3 \in \bullet x \mapsto x^3\} = \{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 8\}$$

$$A \triangleleft \mathfrak{R} = \{x : \mathbb{N} \mid x \mapsto x^3 \in \mathfrak{R} \wedge x \sqsubseteq 4 \bullet x \mapsto x^3\} = \{4 \mapsto 64, 5 \mapsto 125\}$$

$$B \triangleright \mathfrak{R} = \{x : \mathbb{N} \mid x \mapsto x^3 \in \mathfrak{R} \wedge x^3 \sqsubseteq 25 \bullet x \mapsto x^3\} = \{3 \mapsto 27, 4 \mapsto 64, 5 \mapsto 125\}$$

$$\mathfrak{R}(|A|) = \{x : \mathbb{N} \mid x \mapsto x^3 \in \mathfrak{R} \wedge x \leq 4 \bullet x^3\} = \{0, 1, 8, 27\}$$

* Composition

$$X \xleftarrow{R} Y \xleftarrow{S} Z$$

$$R; S = \{x : X, z : Z \mid (\exists y : Y \bullet (x \mapsto y \in S \wedge y \mapsto z \in S \bullet x \mapsto z))\}$$

4) Les fonctions

- une fonction est une relation tel que tous les éléments de l'ensemble de départ sont appliqués dans au plus un couple ordonné de celle-ci.
- Une fonction totale est une fonction pour lequel, le domaine correspond à l'ensemble de départ
- Une fonction partielle est une fonction par laquelle, le domaine est explicitement inclus dans l'ensemble de départ.
- Une fonction injective est une fonction par laquelle à chaque élément du domaine correspond un élément différent de l'image.
- Une fonction subjective est une fonction par laquelle l'image correspond exactement à l'ensemble d'arrivée.
- Une fonction bijective si elle est à la fois injective et subjective.
- Une fonction finie est une fonction dont le cardinal existe.

Notations :

\longrightarrow : Fonction totale

\dashrightarrow : Fonction partielle

$\triangleright \longrightarrow$: Fonction totale injective

$\triangleright \dashrightarrow$: Fonction partielle injective

$\longrightarrow \gg$: Fonction totale subjective

$\dashrightarrow \gg$: Fonction partielle subjective

$\triangleright \longrightarrow \gg$: Fonction bijective

$\dashrightarrow \#$: Fonction finie

$\triangleright \dashrightarrow \#$: Fonction finie injective

On peut déclarer les fonctions par :

- extension $\{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 8, 3 \mapsto 27\}$
- compréhension $\{x : \mathbb{N} \mid x < 4 \wedge x \mapsto x^3 \in F \bullet x \mapsto x^3\}$
- abstraction Lambda $\lambda x : \mathbb{N} \mid x < 4 \wedge x \mapsto x^3 \in F \bullet x^3$

Exemple :

$$\{x : \mathbb{N} \mid x < 4 \bullet x^3 \mapsto x\}(27) = 3$$

$$\lambda x : \mathbb{N} \mid x < 4 \wedge x^3 \mapsto x \bullet x(27) = 3$$

V - Le langage schématique de Z

C'est un langage qui est utilisé pour composer et structurer une série de descriptions mathématiques, il permet de grouper l'ensemble des informations, les encapsuler et de les nommer pour réemploi, en effet, ce langage va nous permettre de présenter des schémas manipulés pour spécifier fortement un logiciel.

Une spécification formelle est composée de deux parties :

- des déclarations globales
- un ou plusieurs schémas

Un schéma se déclare en 3 parties :

- un nom qui identifie le schéma.
- Une partie déclarative qui est peut être vide, qui définit les différentes données membres du schéma.
- Une partie prédictive qui peut être vide, qui énumère les contraintes et les affectations.

Notation :

Nom
Déclarations
Les contraintes et les affectations (une par ligne)

Liaison :

Une liaison est une association entre des noms et des valeurs. Les noms sont des identifications des membres de la partie déclarative d'un schéma tandis que les valeurs sont les expressions associées qui respectent les contraintes d'un schéma.

En Z, on pourra avoir accès à chacun des membres des liaisons à travers un opérateur de sélection. *Exemple* $\ell \bullet x \neq 5$ (x ne possède autre valeur que 5)

Déclaration schématique:

n : la variable dans son état initial

n' : la variable dans son état final

n? : variable paramètre d'entrée du schéma dans son état initial

n! : variable paramètre de sortie dans son état final

Δn : il s'agit d'un raccourci pour déclarer n et n'

Ξ : (Xi) le membre ou schéma qui inclut n ne sera pas modifié

Etude de cas :

Rédaction d'un document de spécification formelle. Le document de spécification formelle contient tout d'abord une introduction formelle au problème à traiter.

→ **Introduction :**

Cette spécification concerne l'enregistrement des passagers à bord d'un avion. Les places ne seront numérotées. Les passagers sont autorisés à embarquer selon la règle du premier arrivé premier servi.

→ **Déclaration de types et de variables :**

Types

[PERSONNE]

Capacité : N

ouiounon ::= oui | non

REPONSE ::= déjàABord | plein

→ **Description de l'état du système :**

→ **Etat du système :**

Avion
àBord : IP Personne
#àBord <= capacité

(IP x : sous ensemble de x)

→ Etat initial du système :

Init
Avion (en se réfère au schéma avion)
$\text{àBord} = \emptyset$

→ : Evolution du système :

ΔAvion
$\text{àBord} : \text{IP PERSONNE}$ $\text{àBord}' : \text{IP PERSONNE}$
$\#\text{àBord} \leq \text{capacité}$ $\#\text{àBord}' \leq \text{capacité}$

→ Les opérations

Embarquer
ΔAvion $P ? : \text{PERSONNE}$
$P ? \notin \text{àBord}$ $\#\text{àBord} < \text{capacité}$ $\text{àBord}' = \text{àBord} \cup \{P ?\}$

Débarquer
ΔAvion $P ? : \text{PERSONNE}$
$P ? \in \text{àBord}$ $\text{àBord}' = \text{àBord} \setminus \{P ?\}$

→ Les interrogations :

ΞAvion
$\text{àBord} : \text{IP PERSONNE}$ $\text{àBord}' : \text{IP PERSONNE}$
$\#\text{àBord} \leq \text{capacité}$ $\#\text{àBord}' \leq \text{capacité}$ $\text{àBord} = \text{àBord}'$

Nombre
ΞAvion $n ! : \mathbb{N}$
$n ! = \# \text{àBord}$

àBord

$\exists \text{Avion}$ $P ? : \text{PERSONNE}$ $\text{reponse} \neq \text{oui ou non}$
$(P ? \in \text{àBord} \wedge \text{reponse} \neq \text{oui}) \vee$ $(P ? \notin \text{àBord} \wedge \text{reponse} \neq \text{non})$

→ **Traitement des erreurs:**

Erreur Embarqués

$\exists \text{Avion}$ $P ? : \text{PERSONNE}$ $\text{rep} ! : \text{REPONSE}$
$(P ? \in \text{àBord} \wedge \# \text{àBord} < \text{capacité} \wedge \text{rep} \neq \text{déjàABord}) \vee$ $(P ? \in \text{àBord} \wedge \# \text{àBord} = \text{capacité} \wedge \text{rep} \neq \text{plein})$

Spécification en Z des fonctions

- 1 - $f' = f \cup \{ x \mapsto y \}$: ajout d'un couple de valeurs à f.
- 2 - $f' = f \oplus \{ x \mapsto y' \}$: modification de $x \mapsto y$ par $x \mapsto y'$.
- 3 - $f' = \{ x \} \triangleleft f$: soustraction au domaine (si on veut supprimer le couple (x,y), il suffit de supprimer x du domaine)

Etude de cas N°2:

Il s'agit d'un exemple simplifié d'une gestion du stock, le niveau de stock est défini pour les produits stockés, il peut exister des articles non stockés.

[ARTICLE] : ensemble des articles stockés ou non.

Magasin

$\text{Stockés} : \text{IP ARTICLE}$ $\text{Niveau} : \text{ARTICLE} \rightarrow \text{IN}$
$\text{dom niveau} = \text{stockés}$

Init

Magasin

$\text{Stockés} = \emptyset$ $\text{Niveau} = \emptyset$

 $\Delta \text{Magasin}$

$\text{Stockés} : \text{IP ARTICLE}$ $\text{Stockés}' : \text{IP ARTICLE}$

EntréeStock

Δ Magasin $a? : \text{ARTICLE}$ $qté? : \text{IN}$
$a? \in \text{Stockés}$ $\text{Niveau}' = \text{Niveau} \oplus a? \mapsto \text{Niveau } a? + qté?$ $\text{Stockés}' = \text{Stockés}$

SortieStock

Δ Magasin $a? : \text{ARTICLE}$ $qté? : \text{IN}$
$a? \in \text{Stockés}$ $\text{Niveau } a? \geq qté?$ $\text{Niveau}' = \text{Niveau} \oplus a? \mapsto \text{Niveau } a? - qté?$ $\text{Stockés}' = \text{Stockés}$

NouvelArticle

Δ Magasin $a? : \text{ARTICLE}$
$a? \notin \text{Stockés}$ $\text{Niveau}' = \text{Niveau} \cup \{a? \mapsto 0\}$ $\text{Stockés}' = \text{Stockés} \cup \{a?\}$

AbandonArticle

Δ Magasin $a? : \text{ARTICLE}$
$a? \notin \text{Stockés}$ $\text{Niveau}' = \{a? \} \triangleleft \text{Niveau}$ $\text{Stockés}' = \text{Stockés} \setminus \{a?\}$

Chapitre 5 : La gestion des projets logiciels

I - Introduction

1) Objectifs

Les objectifs de ce chapitre sont :

- Rendre un étudiant capable de réaliser des activités de gestion de projets logiciels : organisation, estimation du coût, planification, rédaction des documents, gestion de configuration, ... etc.
- S'initier au métier d'informaticien de gestion (gestion de projet, chef de projet)

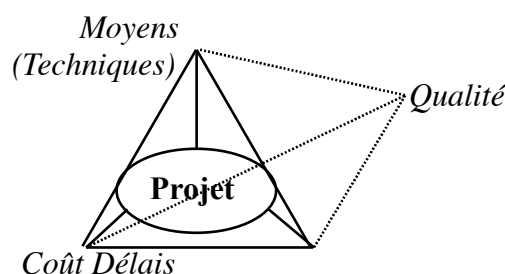
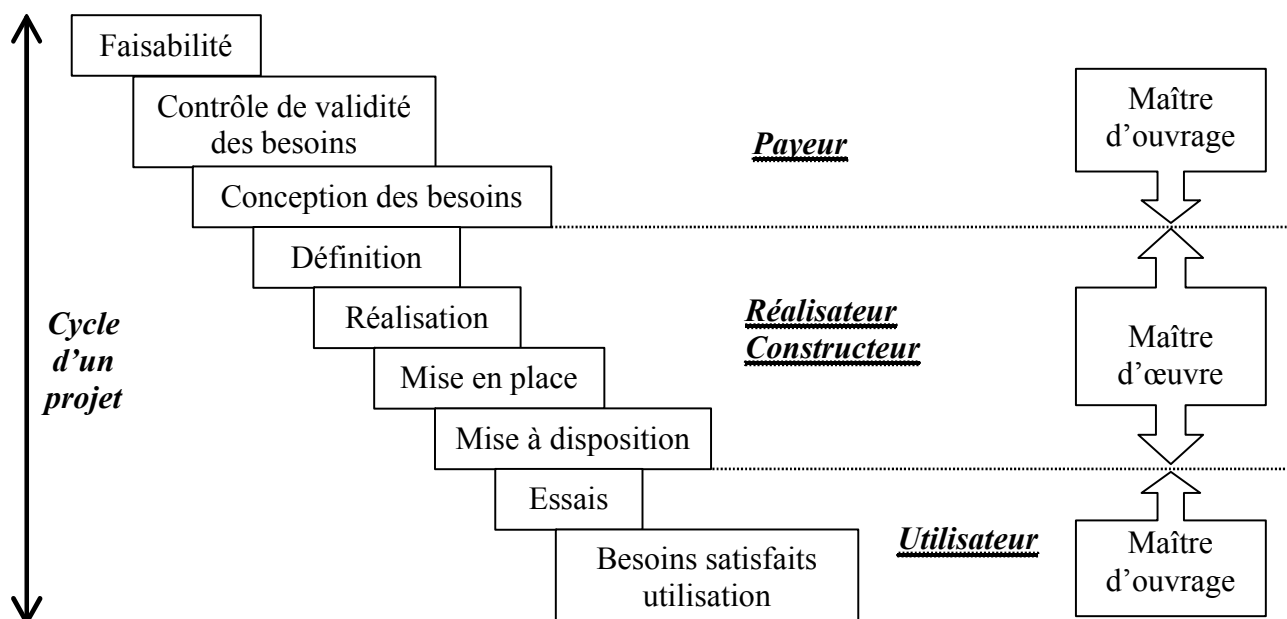
Un projet se distingue par un besoin unique, non répétitif, à une durée déterminée, une activité créatrice, une complexité et une variété d'intervenants qu'il faut coordonner.

2) Définition

Un projet se définit comme une démarche spécifique qui permet de structurer méthodiquement et progressivement une réalité à venir. Un projet est défini et mis en œuvre pour élaborer la réponse aux besoins d'un utilisateur direct ou d'une clientèle et il applique un objectif et des actions à entreprendre avec des ressources données.

Le rôle du chef de projet est de répondre au cahier des charges à la bonne date au coût prévu ou inférieur. Il essaye de trouver le meilleur compromis entre délai et coût.

Cycle d'un projet



II - Cahier des charges

1) Généralités

Le cahier des charges est la responsabilité du client (maître d'ouvrage) ; il doit garantir que son expression traduit bien les besoins initiaux et qu'il soit compréhensible par les tiers.

Le cahier des charges construit une demande de réponses, cette opération se nomme « *appel d'offre* ».

2) Structure d'un cahier des charges**CAHIER DES CHARGES : # nom du projet #**

Référence : # nom de l'entreprise, nom du projet, le nom du document de cahier des charges ...

Date : --/--/----

Version : 1.0

Auteurs :

Objet du document : # description des exigences fonctionnelles et non fonctionnelles pour la conception et la réalisation de l'application et des modalités d'intervention du prestataire.

Clauses juridiques à respecter sous forme d'articles.

Table des Mise à jour du doucement

Version	Date	Objet de la mise à jour
1.0 --/--/----		Document initial

*Sommaire***Parties Pages**

I	xx
II	xx
III	xx
...	xx
IX	xx

I. Objectif de la consultation

- # Préciser l'objet du cahier des charges
- # Préciser le type du marché ou contrat
- # Enumérer les prestations confiées
- # Préciser l'interlocuteur de la direction informatique, son adresse, son téléphone, son mail, ...

II. Présentation de l'entreprise

- # Présenter l'organisation de l'entreprise

III. Présentation de la direction informatique**IV. Présentation générale de l'application****IV. 1 - Description de l'application****IV. 2 - Les acteurs et leurs rôles****IV. 3 - Limites de l'application**

- # Domaines non couverts
- # Exigence non demandée en terme de fonctionnalité, format, contrôle, etc. ...

IV. 4 - Extension potentielle au long terme

- # Autres fonctionnalités à ajouter
- # Autres domaines d'application
- # Autres utilisateurs futures

IV. 5 - Organisation du projet

- # Préciser les structures et les responsabilités : maître d'ouvrage, structure de décision, structure représentative des utilisateurs, équipe du projet

V. Exigences Fonctionnelles et d'Ergonomie/Graphisme

Pour décrire les fonctions à élaborer, on doit faire une description textuelle très détaillée pour les utilisateurs novices et on peut joindre le diagramme de cas d'utilisation ramifié en annexes pour les utilisateurs expérimentés.

Dans cette partie, l'entreprise doit aussi concevoir l'interface utilisateur et pourra offrir une adresse URL comportant la maquette ou le prototype ; aussi l'entreprise doit fixer les règles d'ergonomie et la charte graphique spécifique à l'application

Aussi dans cette partie, on représente les contrats rattachés à l'aspect graphique, on pourra spécifier les formats des images fournis en prestataires et la résolution des images fournis par les prestataires.

VI. Exigences techniques

Exemple : Temps de réponse, Utilisation de la mémoire, maintenance ...

VII. Modalité d'interventions sollicitées**VII. 1 - Contexte d'intervention du candidat retenu**

Domaine de demande de service du prestataire (l'accès à lui fournir dans l'entreprise)

VII. 2 - La démarche de développement

On peut imposer au prestataire d'utiliser une méthode (cycle de vie) ou des outils bien déterminés pour faciliter la maintenance ou conserver la compatibilité dans le système par exemple.

VII. 3 - Echancier

L'entreprise pourra aussi préciser les contraintes de l'échéancier du projet, elle pourra par exemple imposer une date de début du projet, une date limite de fin du projet et les principaux jalons (un jalon est une tâche de durée nulle qui marque le début ou la fin d'une autre tâche).

Exemples de jalon : - réunion avec la maîtrise d'ouvrage

- mise en production sur site pilote

- mise en production sur site général

VIII. Description des prestations attendues

Décrire les prestations attendues de notre candidat final retenu

Prestation : - les résultats de la conception
- Architecture du système sous forme de composants
- Les algorithmes sous forme de pseudo code
- Les structures de données utilisées

Dans cette partie, on décrit ce qui est demandé au candidat retenu de réaliser tout au long du projet, on pourra par exemple, préciser ces prestations par phase du cycle de vie en terme de documents et de livrables. Aussi dans cette partie, on doit préciser des contraintes liées à la garantie de conformité du logiciel, aux spécifications annoncées.

Exemple : suivie des incidents, assurer la correction des logiciels à la suite des anomalies identifiés, ...

IX. Modalités de la consultation

Définir les règles pour répondre à l'autre. Dans cette partie on explicite les modalités à respecter par le prestataire, on doit préciser le cadre de réponse. Aussi, on demande au prestataire de respecter une certaine organisation de leur effort. Un calendrier à respecter qui fixe la durée de l'offre (délai de validité de l'offre).

III - Organisation d'un projet

3 outils à utiliser :

1- **WBS : Work Breakdown Structure** (arbre des tâches)

2- **PBS : Product Breakdown Structure** (arbre des résultats)

3- **OBS : Organisation Breakdown Structure** (Tableau d'affectation des ressources)

PROJET : Réaliser un voyage

T1 - choisir destination

T1.1 + contacter agences de voyages

T1.2 + Consulter les sites web des agences de voyages

T1.3 + Lire les annonces des offres de voyages

T1.4 + Demander l'avis de l'entourage

T2 - Fixer le budget

T2.1 + Trouver les ressources

T3 - Fixer la destination

T4 - Préparer le voyage

T4.1 + Demande VISA

T4.2 + Souscrire une assurance voyage

T4.3 + Préparer passeport

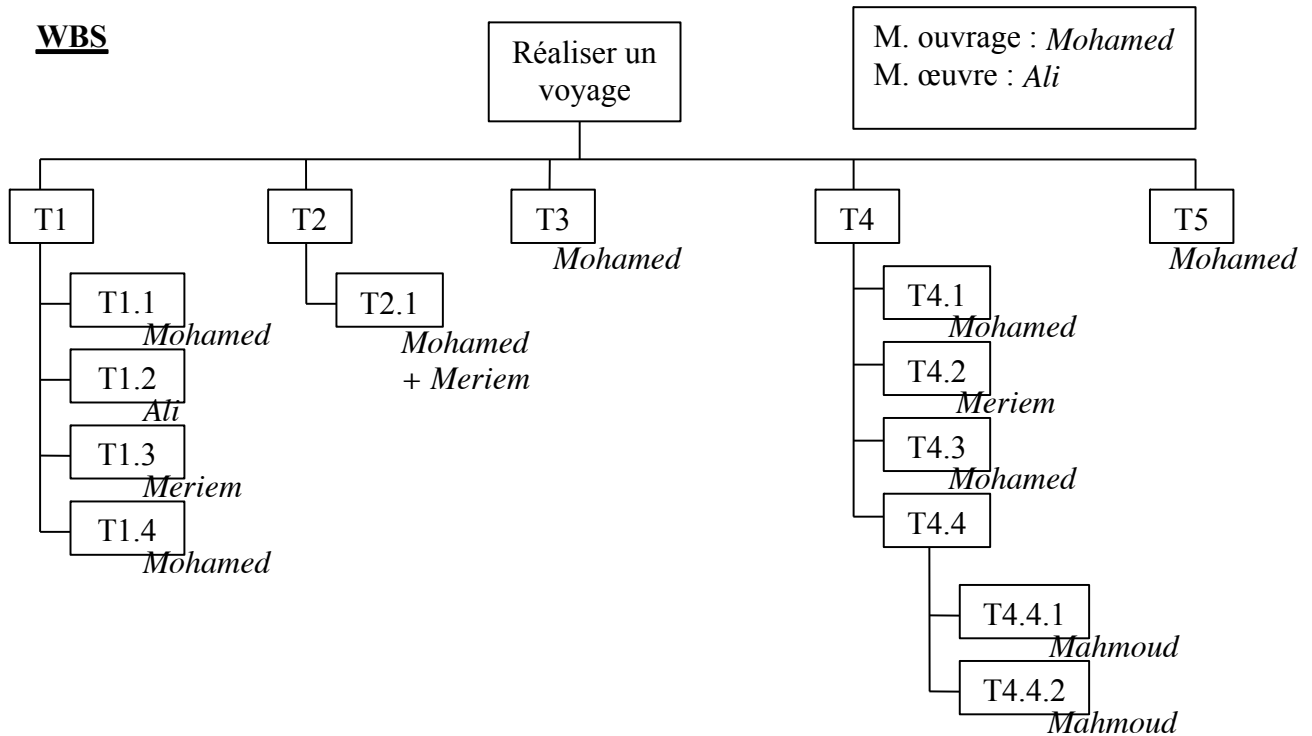
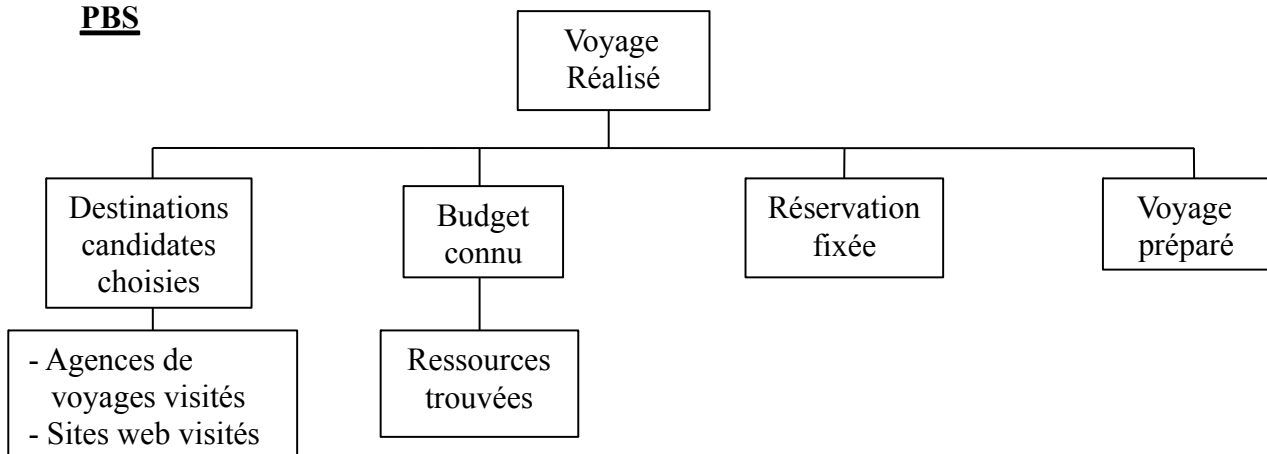
T4.4 + Effectuer réservations

T4.4.1 * Effectuer réservation Agence de voyage

T4.4.2 * Effectuer réservation Billet d'Avion

T5 - Acheter billet

T6 - Voyager

WBS**PBS****OBS**

<i>Ressources</i> <i>Tâches</i>	Mohamed	Meriem	Ali	Mamoud
T1.1 X				
T1.2				X
T1.3		X		
T1.4 X				
T2.1 X		X		
T3 X				
T4.1 X				
T4.2		X		
T4.3 X				
T4.4.1				X
T4.4.2				X
T5 X				

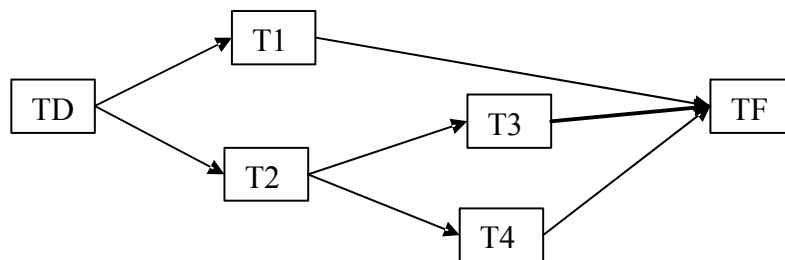
IV - La planification des projets

1) Principes de la planification

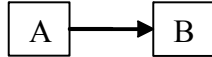

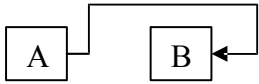
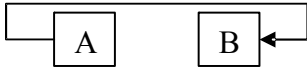
- Ordonnancer
- Prévoir
- Adapter
- Contrôler
- Décider

2) Techniques de la planification

a) PERT



C'est une présentation des tâches et des liaisons. Les différents types de liaison entre tâches sont :

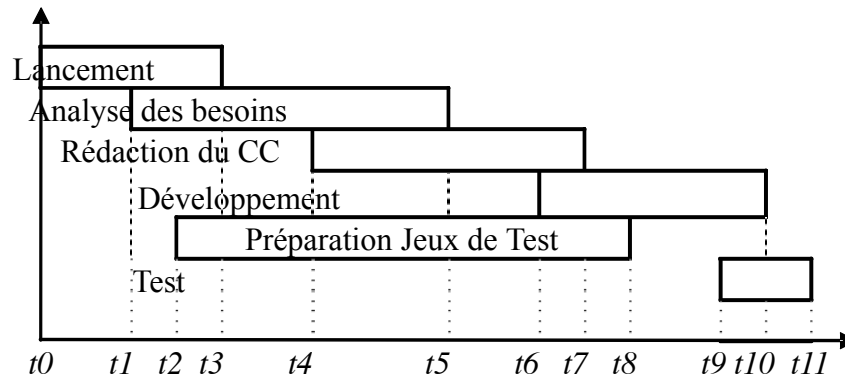
- Fin à début : B ne peut débuter que si A soit terminé. 
- Début à début : B ne peut débuter avant que A soit commencé. 
- Fin à fin : B ne peut être achevé avant que A soit terminé. 
- Début à fin : B ne peut pas se terminer avant que A soit débuté. 

Tâche
Date début
Date fin
Ressources

- ❖ *Date au plus tôt* : c'est la date à laquelle une tâche peut commencer en fonction de dépendances aux autres tâches
- ❖ *Date au plus tard* : c'est la date à laquelle une tâche peut commencer au plus tard sans mettre en cause la date fin du projet.
- ❖ *Marge totale* : c'est l'intervalle de temps pendant lequel une tâche peut être retardée sans affecter la date de fin du projet = **date début au plus tard - date début au plus tôt**
- ❖ *Marge libre* : c'est l'intervalle de temps pendant lequel une tâche peut être retardée sans affecter d'autres tâches = **date début au plus tôt du successeur le plus immédiat - date fin au plus tôt**
- ❖ *Chemin critique* : c'est la suite des tâches ayant une marge totale = 0. Tout retard sur une tâche du chemin critique affecte la date fin du projet.

b) GANTT

C'est un graphe qui représente des tâches en fonction des durées

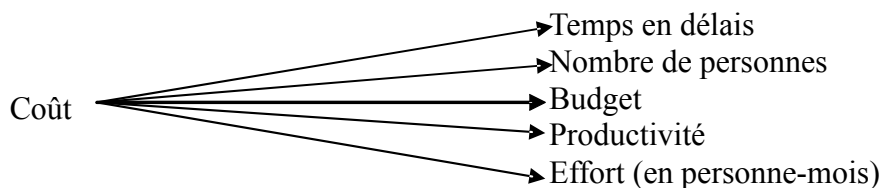
Exemple**V - L'estimation du coût d'un logiciel**

Parmi les problèmes qui faussent l'estimation du coût, on trouve,

Les raisons de sous-traitance :

- l'optimisme
- la négligence des problèmes techniques
- le manque d'expérience
- oublie de la documentation
- sous estimation des tâches annexes
- ...

Le dépassement des délais budgets est une chronique des services informatique, d'où la nécessité des méthodes d'estimation des coûts est apparue.



Au sein de 100 personnes-mois, les personnes et les mois ne sont pas interchangeables car on peut pas spécifier tous les caractéristiques des 100 personnes

1) La méthode par expertise

Il s'agit de faire plusieurs estimations par des experts qui ne se connaissent pas, si les résultats sont concordants ils peuvent être utilisés sinon il est nécessaire d'examiner avec chaque expert les causes des divergences et éventuellement rectifier les estimations en fonction de ces causes.

Généralement un expert doit fournir 3 estimations : une optimiste, une pessimiste et une probable ; on utilise ensuite la méthode des 3 valeurs pour calculer les estimations

$$= \frac{E_{\text{optimiste}} + 4 * E_{\text{probable}} + E_{\text{pessimiste}}}{6}$$

2) La méthode par analogie

C'est une méthode qui se base sur des analogies faites au sein de l'entreprise ou en dehors de l'entreprise, l'analogie peut être même avec des composants. On majore ou on réduit les coûts des projets analogues suivant les spécificités des projets pour trouver une estimation adéquate.

3) La méthode par fonction

Cette méthode est basée sur les principes suivants

- estimer le nombre de fonctions par type
- estimer le nombre de lignes de code (LOC *Lignes Of Code*) pour chaque fonction
- utiliser la table de productivité
- faire la somme

Type fonction	Productivité en PM/K-LOC
Mathématiques 6	PM/K-LOC
Edition 8	PM/K-LOC
Logique 12	PM/K-LOC

Traitement de signaux/ Contrôle processus	20 PM/K-LOC
Contrôle temps réel	40 PM/K-LOC

Exemple :

Si on suppose que le nombre de lignes de codes d'une fonction mathématique 2 KLOC, pour une fonction d'édition 2 KLOC, pour une fonction logique 2 KLOC, 20 KLOC pour une fonction de contrôle de processus

$$\text{Effort} = (6*2) + (8*2) + (12*5) + (20*20) = 488 \text{ PM}$$

4) La méthode de répartition proportionnelle

Elle est adaptée aux projets décomposés en étapes et surtout pour les phases et les tâches classiques : étude préalable, étude détaillée, étude technique, réalisation et mise en œuvre.

Seule l'étude préalable fait l'objet d'une quantification analytique claire. Elle est divisée en 3 phases : observation, conception et appréciation ; cette méthode tient compte aussi des charges complémentaires annexes correspondant à une tâche suivante (Encadrement du projet, Recette, Documentation utilisateur). Les tableaux suivants indiquent des ratios à ajuster selon l'expertise du chef du projet.

Etapes Ratios	
Etude préalable	10% du projet
Etude détaillée	20% - 30% du projet
Etude technique	5% - 10% d la charge de l'étude détaillée
Réalisation	2 fois la charge de l'étude détaillée
Mise en œuvre	30% - 40% d la charge de réalisation

Etude préalable	Ratios
Observation	30% - 40%
Conception	50% - 60%
Appréciation 10%	

Charges complémentaires :

Tâches Ratios	
Encadrement (étapes de réalisation)	20% de la charge de réalisation
Encadrement (autres étapes)	10% de la charge de l'étape
Recette	20% de la charge de réalisation
Documentation utilisateur	5% de la charge de réalisation

5) Modèle COCOMO: CONstructive COSt MOdel (Barry Boehm - Software Engineering Economies)

3 modes de développement :

- Mode organique : qui couvre les projets faciles qu'on a l'habitude de réaliser.

- Mode semi détaché : qui couvre les projets moyennement complexes.

- Mode contraint (imbriqué) : qui couvre les projets complexes.

Pour chaque mode, COCOMO propose 2 équations : une pour calculer l'effort de développement et une pour calculer le temps de développement.

a) COCOMO de base

Mode organique	Mode semi détaché	Mode contraint
$\text{Effort}_{\text{DEV}} = 2.4 (\text{KLOC})^{1.05}$	$\text{Effort}_{\text{DEV}} = 3 (\text{KLOC})^{1.12}$	$\text{Effort}_{\text{DEV}} = 3.6 (\text{KLOC})^{1.20}$
$\text{Temps}_{\text{DEV}} = 2.5 (\text{Effort}_{\text{DEV}})^{0.38}$	$\text{Temps}_{\text{DEV}} = 2.5 (\text{Effort}_{\text{DEV}})^{0.35}$	$\text{Temps}_{\text{DEV}} = 2.5 (\text{Effort}_{\text{DEV}})^{0.32}$

$$\text{Productivité} = \frac{\text{Effort}}{\text{KLOC}}$$

$$\text{FSP (Full Software Person = Nombre de personnel Moyen)} = \frac{\text{Effort}_{\text{DEV}}}{\text{Temps}_{\text{DEV}}}$$

$$\text{ACT (Annual Change Trafic = Taux des Instructions Modifiées)} = \frac{\text{nombre des instructions modifiées}}{\text{nombre total des instructions}}$$

$$\text{Effort}_{\text{AM}} = \text{ACT} * \text{Effort}_{\text{DEV}}$$

$$\text{FSP}_{\text{AM}} = \text{Effort}_{\text{AM}} / 12$$

Rémarque :

3 facteurs seulement ont été considérés par COCOMO de base

- la taille du projet
- le mode de développement
- ACT

b) COCOMO intermédiaire

COCOMO intermédiaire est plus détaillée que COCOMO de base, en fait il a introduit 15 autres facteurs multiplicatifs du coût regroupés dans 4 catégories

i) catégorie du produit

RELY : un facteur lié à la fiabilité du produit

CPLX : un facteur lié à la complexité du produit

DATA : un facteur lié à la taille de la BD du produit

ii) catégorie du matériel

TIME : un facteur qui exprime la contrainte temps d'exécution

STOR : un facteur qui exprime la contrainte mémoire

VIRT : un facteur qui exprime la volatilité de la machine virtuelle

TURN : un facteur qui exprime le temps de finition d'un processus

iii) catégorie du personnel

ACAP : un facteur qui exprime la capacité des analystes

PCAP : un facteur qui exprime la capacité des programmeurs

AEXP : un facteur qui exprime l'expérience des analystes

LEXP : un facteur qui exprime l'expérience sur les langages

VEXP : un facteur qui exprime l'expérience sur les machines virtuelles

iv) catégorie du projet

MODP : un facteur qui exprime l'utilisation des techniques de programmation moderne

TOOL : un facteur qui exprime l'utilisation des outils

SCED : un facteur qui exprime la contrainte temps de développement

$$\text{EAF (Effort Adjustment Factor)} = \prod_{i=1}^{15} f_i$$

Mode organique	Mode semi détaché	Mode constraint
Effort _{DEV} = EAF * 3.2 (KLOC) ^{1.05} Temps _{DEV} = 2.5 (Effort _{DEV}) ^{0.38}	Effort _{DEV} = EAF * 3 (KLOC) ^{1.12} Temps _{DEV} = 2.5 (Effort _{DEV}) ^{0.35}	Effort _{DEV} = EAF * 2.8 (KLOC) ^{1.20} Temps _{DEV} = 2.5 (Effort _{DEV}) ^{0.32}

Vers la fin des années 90, un nouveau model avait son apparition COCOMO2 , il propose 3 modèles d'estimation du coût :

- le modèle d'application composition : qui est utilisé essentiellement pour le prototype
- le modèle early design (conception préliminaire)
- le modèle cost-architecture

Chapitre 6 : La Normalisation

I - Définitions

Les normes sont des apports documentés contenant des spécifications techniques ou autres critères précis destinés à être utilisés systématiquement en tant que règle ligne directrice ou définition des caractéristiques pour assurer que des matériaux, produits, processus et services sont aptes à leurs envoies.

Les normes internationales attribuent ainsi à nous simplifier la vie et à accroître la fiabilité et l'efficacité des biens et des services que nous utilisons.

Exemple : carte de crédit : le format de ces cartes est dérivé de la norme ISO qui a défini des caractéristiques tel que l'épaisseur optimale (0,76 millimètres) cela signifie que les cartes pourront être utilisés dans le monde entier.

II - La norme ISO

C'est une organisation internationale de normalisation créée en 1947. C'est une fédération mondiale d'organismes nationaux de normalisation. Un organisme par pays (INORPI, en Tunisie). Elle a comme mission de favoriser le développement de la normalisation et des activités connexes dans le monde e vue de faciliter, entre les nations, les échanges des biens et des services et de développer la coopération dans les domaines intellectuels, scientifiques, techniques et économiques. Les travaux d'ISO aboutissent à des accords internationaux qui sont publiés sous forme de normes internationales. On a plus que 12000 normes publiées depuis 1947.

Une norme ISO est représentée par un document normatif est élaboré selon des procédures approuvées par les membres d'ISO.

III - Etapes pour la certification ISO 9000

ISO 9000 traduit la démarche qualité au niveau de l'entreprise, elle apporte :

- en interne (à l'échelle nationale) : l'amélioration de la compétitivité et la diminution des coûts.
- à l'extérieur (à l'échelle internationale) : la réponse aux exigences des donneurs d'ordre qui imposent de plus en plus la certification ISO 9000 à leurs fournisseurs.

L'implication de la normalisation ISO 9000 importe à l'entreprise :

- une amélioration de sa compétitivité
- Une réponse aux demandes du marché
- Une démarche fédératrice pour l'ensemble des domaines
- Un axe de communication externe

Pour mettre en œuvre une norme ISO 9000 au sein d'une organisation on doit tout d'abord réaliser un diagnostic de l'organisation en auditant les fonctions principales de l'organisation, ce diagnostic pourra apparaître un certain nombre de non conformités. Le rôle du responsable qualité est de prendre en charge ce côté formel ; le responsable qualité va veiller à établir un plan de travail traditionnellement, les points nécessaires par action sont :

- La gestion documentaire
- La revue du contrat (c'est ce qu'on est en mesure de pouvoir répondre à une demande du client à juste prix ?)
- La gestion des produits non-conformes
- La traçabilité

Parallèlement, il faut s'intéresser pour tous les processus de l'entreprise de la prise de commandes jusqu'à la livraison des produits et du SAV

Les intérêts recherchés sont :

- Les fonctions du personnel sont bien définies (qui fait quoi ?)
- Le processus est établi de la commande jusqu'à la livraison
- Les procédures sont établis, il ne reste qu'à les formaliser (quoi faire ? pourquoi ?)
- La circulation des documents précisés
- Les points critiques apparaissent

IV - Normes IEEE

Le terme « norme » désigne un ensemble de règles à suivre dans la réalisation d'un produit. Les normes IEEE dans la génie-logiciel couvre un ensemble de 37 normes s'intéressant aux processus, produits et aux techniques de base de génie-logiciel, cet ensemble a comme caractéristiques d'être ouvert à normes internationales ISO.

IEEE std 12207.0 : qui s'intéresse au processus du cycle de vie d'un logiciel. Cette norme s'est intéressé à fixer la signification de certains termes et choisir les processus prioritaires. Elle peut être considérée comme un dictionnaire et une liste de vérification pour la gestion des projets.

IEEE std 1362 : guide pour la rédaction du document principe d'opération, pour faciliter l'écriture et l'étude des opportunités

IEEE std 830 : recommandée pour les spécifications des exigences du logiciel, cette norme pourrait être intégrée avec les approches de validation avec prototype.

IEEE std 1028 : pour appliquer la validation et la vérification. C'est une norme très facile à adapter à tous les problèmes et qui permet en u très peu de temps d'avoir cette norme.