Chapitre 1 : INTRODUCTION AU GENIE LOGICIEL

1-Introduction :

L'informatisation a évolué en fonction des besoins et des objectifs des entreprise d'une part, du progrès technologique d'autre part. Face à l'évolution les systèmes informatiques se sont modifiés dans leur approche et leurs fonctionnalités. En premier lieu l'informatisation de certaines tâches répétitives simples a été faite dans le but d'améliorer les délais d'obtention des résultats et les résultats eux-mêmes. En suite, il a été possible d'intégrer dans un même système de différentes fonctions d'une entreprise et établir ainsi des communications entre les sous systèmes. Ceci pouvant être repartis sur des sites dispersés. Les coûts informatiques sont importants pour leurs plus grande part à la maintenance des logiciels (jusqu'à 80 °/° du total). La complexité des systèmes informatiques et la croissance des coûts de production ont justifié l'étude et le développement de méthodes et d'outils ; afin d'apporter des aides et des normes de production travaux dits du génie logiciel.

2- Définition d'un logiciel :

C'est un ensemble de programmes, de procédés et des règles et éventuellement de la documentation relatifs au fonctionnement d'un ensemble de traitements de l'information. En anglais, on dit le **« SOFTWARE ».**

3- Définition du génie :

C'est un ensemble de connaissances raisonnées scientifiques et des moyens appropriés concernant la conception, la mise en œuvre, et les applications des méthodes et outils, propres à un domaine d'activité.

4- Génie informatique :

Conception, réalisation et validation des systèmes informatiques.

5- Génie logiciel :

C'est l'ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi.
On peut dire aussi qu'un génie logiciel est l'art de spécifier, de concevoir, de réaliser, et de faire évoluer, avec des moyens et dans des délais raisonnables, des programmes, des documentations et des procédures de qualité en vue d'utiliser un ordinateur pour résoudre certain problèmes .

6- Atelier du génie logiciel (AGL) :

Catégorie de logiciels offrant un environnement complet de développement de logiciel en équipe. En anglais ; computer aided software enginneering(CASE).

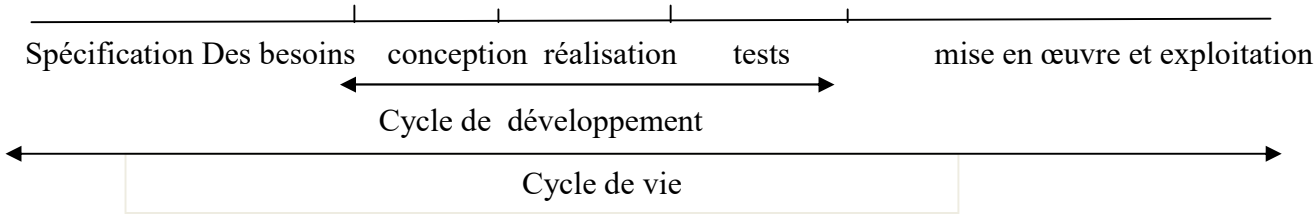
- Les AGL sont apparus dans les années 80 pour faciliter le développement de logiciels en équipes. Ces logiciels disposent d'un ensemble d'outils plus ou moins complets de gestion et d'automatisation de la production de logiciels. Les grandes fonctions sont :
    - La conception des différents points du cahier des charges ;
    - La planification des tâches de développement ;
    - La préparation de la documentation ;
    - La programmation des différents modules du logiciel.

Tout logiciel possède un cycle de vie et un cycle de développement.

7- Cycle de vie d'un logiciel : (software life cycle) C'est l'ensemble des phases du cycle de développement du logiciel précédé par la phase de spécification et suivi de la phase d'exploitation.

8- Cycle de développement d'un logiciel (software développement cycle) : ensemble d'activités mises en œuvre dans un ordre donné pour réaliser un logiciel.

Voici la représentation graphique du cycle de vie en mettant en évidence le cycle de développement.



9- Exploitation : (opération) dernière phase du cycle de vie du logiciel, se composant de :

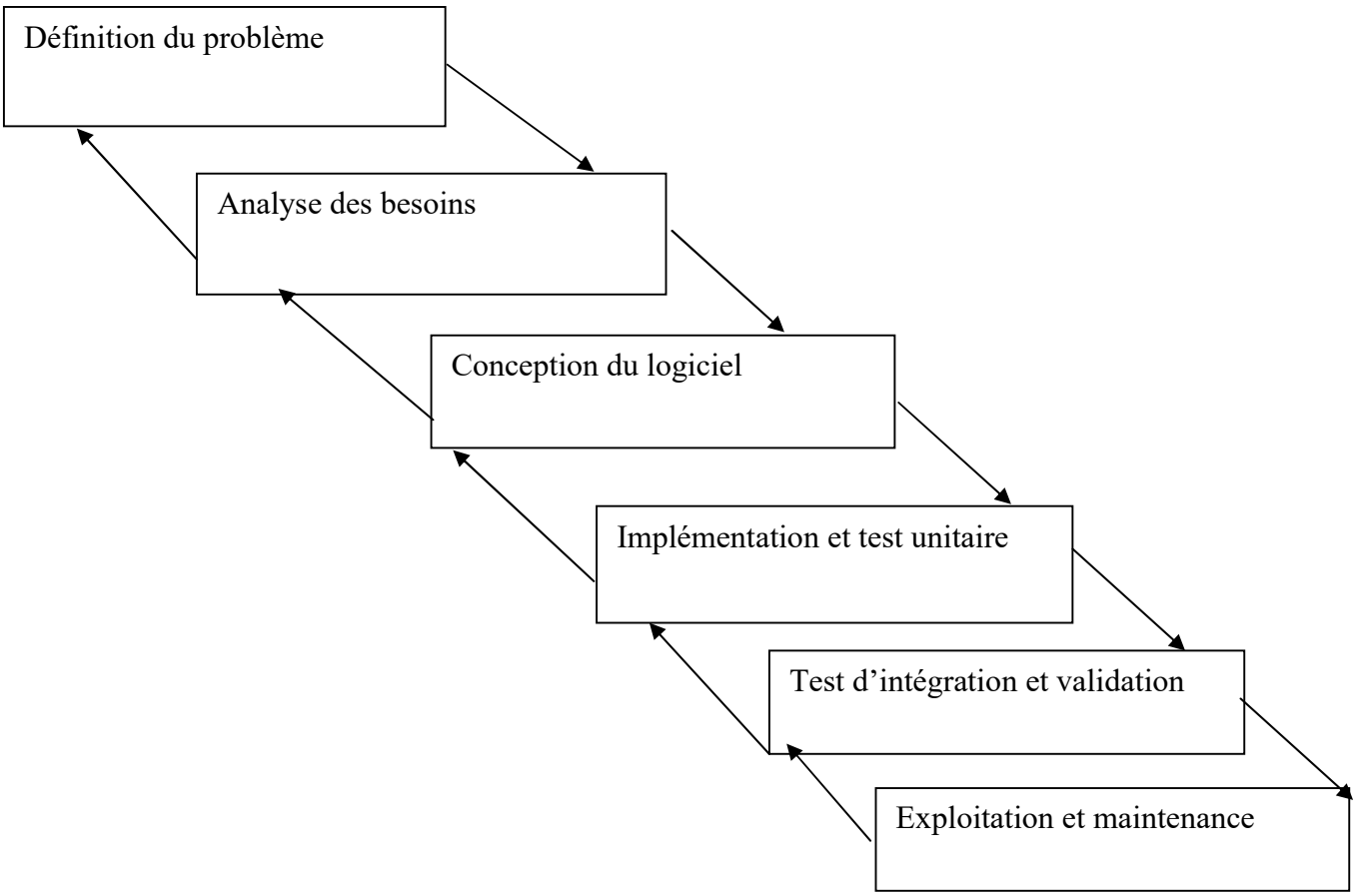
- La mise en œuvre du logiciel (installation, formation, lancement en parallèle avec le manuel) ;
- L'utilisation du logiciel ;
- La maintenance du logiciel ;
- L'adaptation du logiciel.

Voici quelques exemples de modèles de cycles de développement :

- Le cycle de vie traditionnel ;
- Le cycle de vie en V ;
- Le cycle de vie tridimensionnel.
- **Le cycle de vie traditionnel :**

Le cycle de vie traditionnel a été introduit par ROYCE en 1970. C'est une succession d'étapes allant de 5 à 7 étapes. Il possède plusieurs variantes avec ou sans rétroaction. Ce modèle a reçu un accueil enthousiaste de la part des chefs de projets qui vivent là un moyen de rendre le

processus de développement plus visible. Du fait de la descente en cascade d'une étape à l'autre, ce modèle est dit de cascade.



- **Avantage :**

  Simplicité.

- **Inconvénients :**

- On s'est rapidement aperçu qu'il n'était approprié qu'a certaines classes de projets logiciels. Les réalités du développement du logiciel s'accordaient mal avec les activités du modèle en cascade. Le processus logiciel est variable et complexe et l'on ne peut décrire en utilisant un modèle simple comme celui de la cascade.
- Notion de système ignore ce qui implique qu'il n'est pas approprie pour les logiciels complexes (nombre d'application important).

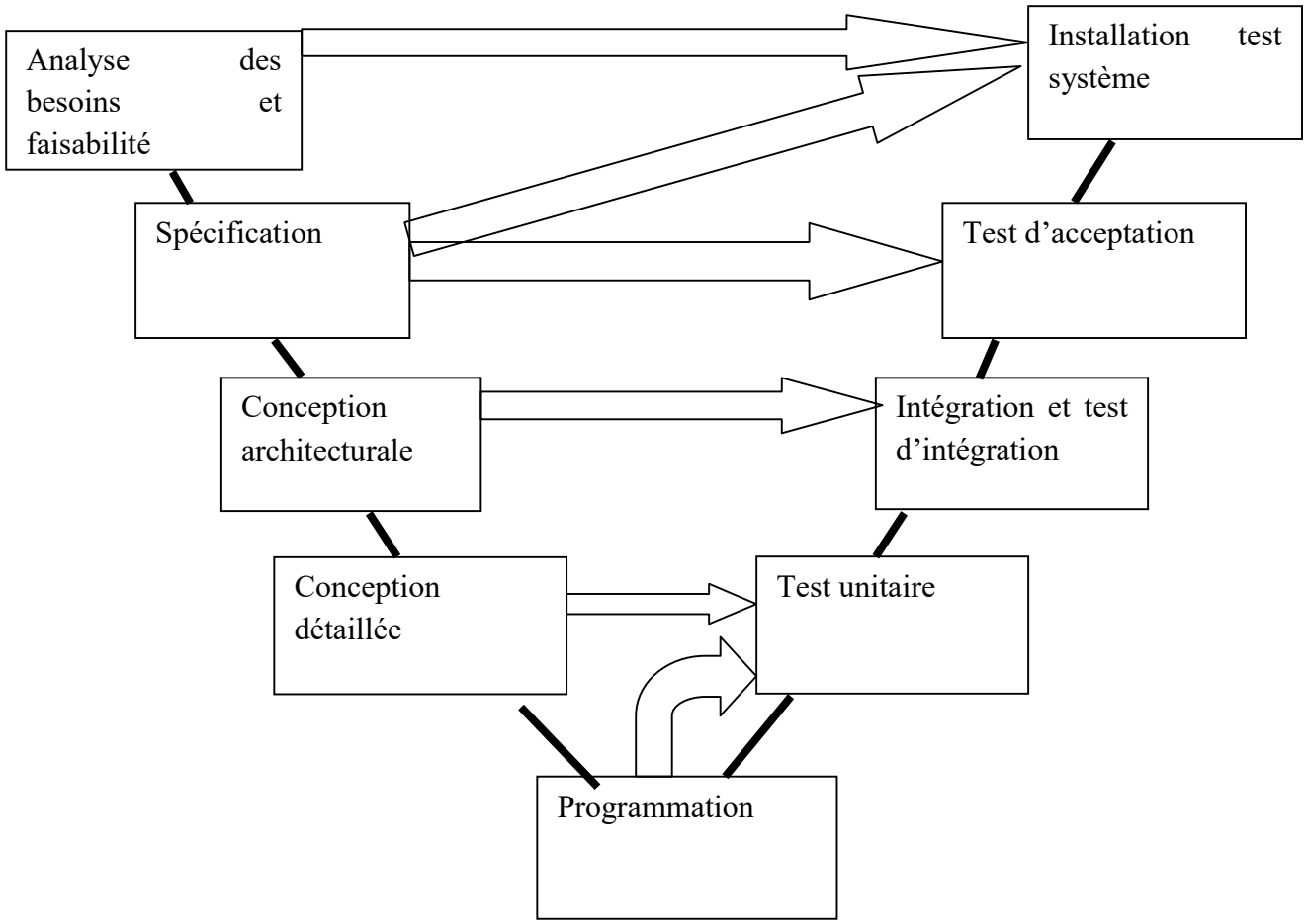
- Les étapes sont chronologique séquentielles (sans retour arrière dans la variante simplifiée).

- Le cycle de vie en v :

- **Définition :**

C'est une variante du cycle de vie en cascade. Il introduit la notion de système et de composant (sous système, module ou partie d'un logiciel).

- **Schéma :**



Ce modèle rend explicite le fait que les premières étapes du développement, qui ont trait surtout à la construction du logiciel, doivent préparer les dernières étapes, qui font intervenir essentiellement l'activité de validation et de vérification.

Il y a deux sortes de dépendances entre étapes :

- Celle matérialisée par des traits gras obliques, qui correspondent à l'enchaînement et à l'interaction éventuelle du modèle en cascade ; les étapes se déroulent donc séquentiellement en suivant le V de gauche à droite ;

- Celles matérialisées par les flèches, qui représentent le fait qu'une partie des résultats de l'étape de départ est utilisée directement par l'étape d'arrivée ; par

exemple à l'issue de la conception architecturale, le protocole d'intégration et les jeux de test d'intégration doivent être complètement décrits.

- **Avantage :**

. Le principe de ce modèle est qu'avec toute décomposition doit être décrite la recomposition, et que toute description d'un composant est accompagnée des tests qui permettront de s'assurer qu'il correspond à sa description. Ce principe évite le problème bien connu de la spécification de logiciel : on annonce une propriété qu'il est impossible de vérifier objectivement une fois le logiciel réalisé.

. Plus grande **maitrise** du développement de systèmes complexes ;

. Contrôler mieux les étapes par une **hiérarchie de tests** ;

. Différencie la **validation** (vérifier que l'on construit ce qu'il faut) de la **vérification** (vérifier que le système est bien construit).

- **Inconvénient :**

La validation par rapport aux besoins intervient trop tard dans le cycle. Il est coûteux de constater l'inadéquation du système seulement à la fin de la réalisation.

- Le modèle tridimensionnel :

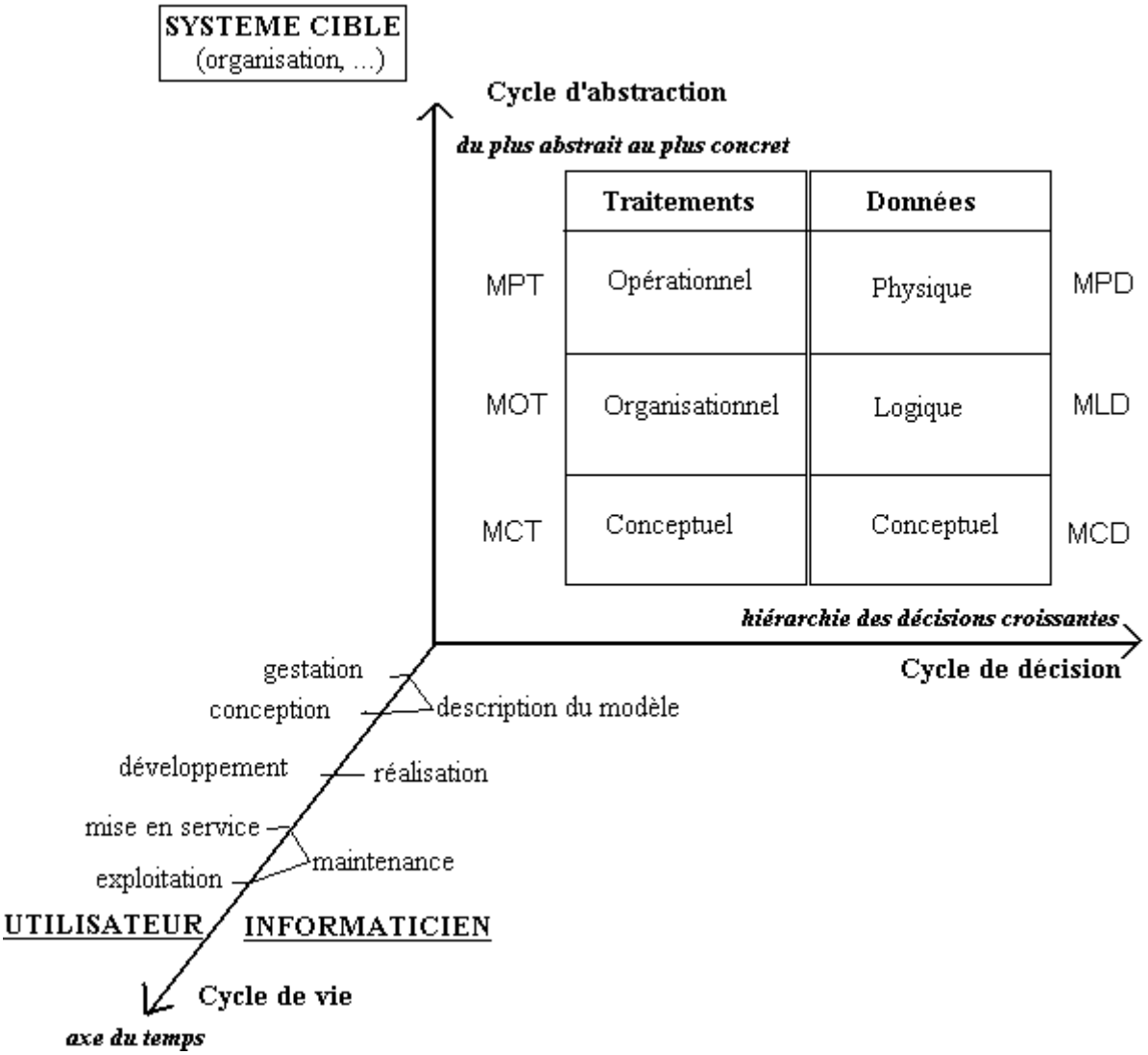
Ce modèle a été introduit par Merise en 1983.

Il s'intéresse aux logiciels de gestion et plus exactement aux systèmes d'information (SI).

Il positionne le développement d'un SI par rapport à trois axes et il considère la conception du SI comme constamment et simultanément guidée par ces trois axes.

- Un axe qui décrit le cycle de vie du SI et qui décrit les principales périodes à l'issue desquelles le SI subit des bouleversements importants : naissance du SI, changement organisationnel et changement technologique ;
- Un axe qui décrit le cycle d'abstraction du SI, c'est à dire, les différents niveaux de description, allant du niveau conceptuel au niveau physique, en passant par le niveau organisationnel et le niveau logique ;
- Un axe qui décrit le cycle de décision comportant une hiérarchie de décisions croissantes.
- **MERISE** :
- M : Méthode ; E : Etude ; R : Réalisation ; I : Informatique ; S : Sous ; E : Ensemble.

Schématisation :



MCD (modèle conceptuel des données)

MCT (modèle conceptuel des traitements)

MOT (Modèle organisationnel des traitements)

MLD (modèle logique des données)

MPT (modèle Opérationnel (ou physique) des traitements)

MPD (modèle physique des données)

- **Avantage:**

L'axe qui décrit le cycle de vie du SI permet de planifier l'évolution du SI et d'organiser son changement et son évolution.

L'axe qui décrit le cycle d'abstraction permet d'assurer une meilleure portabilité et une plus grande évolutivité du SI par l'introduction d'une «indépendance » entre les différents niveaux.

- **Inconvénients :**

Sémantique imprécise du contenu de chaque plan forme par une paire d'axes.

10- Critères de choix d'un cycle de vie :

Il n'existe pas de cycle de vie à priori meilleur en toute circonstance.

Les critères de choix sont :

1. L'environnement du processus du développement
   - Politique de l'entreprise ;
   - Compétences internes ;
   - les expériences intérieures.
2. La complexité du produit ;
3. L'environnement du produit
   - Technique (matériel, réseaux….)
   - Applicatif (bd existante, application existante)
4. Les risques : l'expérience montre qu'il est souvent plus aisé de s'exprimer sur les risques d'échecs que sur les facteurs de réussite.

11. qualités d'un logiciel :

11. 1) notion de qualité : d'après Larousse : « manière d'être, bonne ou mauvaise, de quelque choses ». ISO (international standard d'organisation) aptitude d'un produit ou d'un service à satisfaire les besoins des utilisateurs ; la qualité n'est pas une idée simple mais une notion à plusieurs facettes : les facteurs de qualité.

11. 2) facteurs de qualité :

11.2.1. Facteurs de développement :

a) Validité : aptitude d'un produit logiciel à réaliser exactement les tâches définies dans sa spécification.

b) Fiabilité : aptitude d'un produit logiciel à fonctionner sans incidents comme le précise sa spécification.

c) Efficacité : aptitude d'un produit logiciel à bien utiliser les ressources mémoire, périphériques……

d) Facilité d'utilisation : facilité avec laquelle les utilisateurs peuvent apprendre comment utiliser le logiciel (nécessité d'une interface utilisateur appropriée).

11.2.2. Les facteurs d'évolution :

a) Maintenabilité: aptitudes d'un logiciel à être mis à jour sans trop de coût.

b) Flexibilité : c'est la possibilité d'opérer des modifications dans l'ergonomie du logiciel et dans l'architecture de ses programmes et données.

c) Testabilité : est la possibilité d'opérer des tests sur le logiciel et de les interpréter en matière de justesse des résultats obtenus.

11.2.3. Les facteurs de transition :

a) Portabilité : exprime le fait de pouvoir rendre un logiciel opérationnel dans un environnement matériel et /ou logiciel autre que celui pour lequel il a été conçu.

b) Réutilisabilités : est l'objectif selon lequel le logiciel doit fournir en lui-même un élément réutilisable ou un ensemble d'éléments logiciels réutilisables **.**

c) Interopérabilité **:** c'est la possibilité de couplage ou d'échange d'information avec d'autres logiciels.

12- Les caractéristiques internes d'un logiciel :

a. Audiabilité : degré de conformité avec les standards et les normes internationales.

b. Précision : degré de précision des traitements et contrôles.

c. Généralité de la communication : degré de standardisation des interfaces.

d. Efficacité dans l'exécution : temps d'exécution des programmes.

e. Extensibilité : degré avec lequel l'architecture du logiciel et de ses données peuvent évoluer.

f. Indépendance du matériel : degré de couplage du logiciel avec le matériel sur lequel il a été développé.

g. Modularité : degré d'Independence des composants du logiciel.

h. Sécurité : les mécanismes de contrôle et de protection des programmes et des données du logiciel.

i. Auto –documentation : degré avec lequel le code source fournit une documentation (les commentaires).

j. Simplicité : degré de faciliter de compréhension du logiciel

13- Spécifications exigées d'un logiciel :

Avant le démarrage d'un projet logiciel, des spécifications sont exigées, ces spécifications impliquent négociation, définition des besoins, faisabilité technique et commercial lors de cette étapes, nous devons clarifier ;

- L'estimation des coûts ;
- L'échéancier du projet ;
- Les phases du cycle de vie ;
- Les parties à fabriquer et celles à acheter ;
- Les ressources nécessaires ;
- Les méthodes et outils ;
- Les normes à respecter ;
- Les responsables des équipes ;
- Le maître d'ouvrage en spécifiant les caractéristiques des utilisateurs.

14- Caractéristiques des utilisateurs :

- connaissance ou non de l'informatique ;
- expérience de l'application ;

- utilisateurs réguliers et /ou occasionnels….

1- **Conception d'un logiciel:**

Concevoir un logiciel est un processus créatif qui demande de la perspicacité. Une conception finale est généralement obtenue par un processus itératif à partir de la conception préliminaire.

Une bonne conception est facile à réaliser et à maintenir, facile à comprendre et fiable. Bien qu'il peut fonctionner correctement, un système mal conçu sera souvent coûteux à maintenir, difficile à tester et peu fiable. La phase de conception est donc la phase la plus cruciale du processus de développement d'un logiciel.

Jusqu'à une période très récente, la conception était, pour une grande part, un processus empirique, étant donné un ensemble de besoins, exprimés le plus souvent dans un langage naturel, une conception informelle était proposée, souvent sous forme d'un organigramme. Le codage pouvait alors commencer et la conception initiale était modifiée au fur et à mesure de la réalisation du système. Une fois la réalisation terminée, les spécifications d'origines constituent une description totalement inadéquate du système. Cette approche de conception à provoqué un échec spectaculaire et coûteux pour de nombreux projets. Nous savons actuellement que les organigrammes ne sont pas un moyen adéquat pour formuler une conception .On reconnaît qu'une spécification précise, mais pas nécessairement formelle, est une partie essentielle du processus de conception, et que la conception d'un logiciel est une activité itérative, à plusieurs niveaux. En conséquent un certains nombre de notation de conception, plus adaptées que les organigrammes sont apparu citons par exemples : le dictionnaire de données, le diagramme des flux, le modèle entité association, les réseaux de pétri…

Le concepteur d'un logiciel doit utiliser des besoins pour mettre au point un système qui satisfasse ces besoins. Cette transformation s'accomplit en un certain nombre d'étapes :

Etape 1 : Les sous systèmes constituant ce système doivent être identifiés.

Etape 2 : Chaque sous système doit être décomposé en un certains nombre de composants. La spécification de ces sous systèmes consistera à définir les opérations de ces composants

Etape 3 : Chaque programme doit alors être conçu en termes de composant (module) en interactions.

Etape 4 : Chaque composant (module) peut alors être décomposé dans une hiérarchie de sous composants.

Etape 5 : A un certain stade de ce processus, les algorithmes utilisés dans chaque composant doivent être détaillés.

Il n'existe pas de critères définitifs permettant de définir une bonne conception. Une bonne conception facilite la maintenance et le coût des changements à apporter au système est minimal.

Cela signifie que la conception initiale doit être facile à comprendre, et que l'effet des changements doit rester localiser. Il est possible d'atteindre ces objectifs lorsque la conception apporte à la fois un haut degré de cohésion et un couplage faible.

2. **Modularité :** Un module (unité de traitement) est un ensemble de traitements étroitement liées, on l'appelle aussi composant du logiciel.
3. **Cohésion :** On dit qu'une unité de programmes fait preuve d'un haut degré de cohésion si les éléments la constituant remplissent des fonctions très proches. Cela signifie que chaque élément de cette unité de programme doit être essentiel pour que cette unité remplisse son rôle.

4. **Couplage :** Un couplage est lié à la cohésion .C'est une indication de la force des connexions entre unités. Des systèmes à couplage fort ont des connexions fortes entre des unités qui dépendent les unes des autres, alors que les systèmes à couplages faibles sont constitués d'unités qui sont indépendantes ou presque.

Remarque : L'avantage des systèmes à forte cohésion et à couplage faible est qu'il est possible de remplacer une unité quelconque par unité équivalente avec peu ou pas de changement dans les autres unités du système. Ceci est également important pendant le processus de conception.
La conception d'un logiciel est une tâche créative, mais aussi la plus difficile, cette difficulté appelle donc une réponse méthodologique. Il existe un grand nombre de méthodes de conception par peut de classe comme suit:

2- Classement des méthodes de conception : (des méthodes structurées aux méthodes orientées objet)

a. **Conception fonctionnelle descendante:** Pour ce type de méthode, Le système est conçu d'un point de vue fonctionnel en commençant au niveau le plus général et en descendant progressivement vers la conception détaillée. Cette méthode est appelée conception structurée.
b. **Conception orienté données :** Cette méthode part du principe que la structure d'un logiciel doit refléter la structure des données traitées par ce logiciel. En conséquence, la conception est directement dérivée de l'analyse des données en entrée et en sortie.
c. **La conception orienté objet :** le système est vue comme une collection d'objet communiquant entre eux par messages.