d'un **processus logiciel**, où le logiciel nait, évolue, et meurt (être rejeté) à une certaine phase. Par cela, l'objectif du GL se résumera encore dans deux points :

- La définition de ce processus logiciel
- La gestion de ce processus logiciel

Dans ces objectifs le GL propose ce qui est connue par **Approches de développement de logiciel**, et **Modèle de Gestion de projet Logiciel**.

a) Les approches de développement : une approche de développement est une démarche ou une politique (théorique) permettant de définir les étapes à suivre pour développer un logiciel. Il y'a une variété d'approches :
   - L'approche de la cascade
   - L'approche de prototypage
   - L'approche de la programmation exploratoire
   - L'approche de la réutilisabilité ;
   - L'approche des transformations formelles

b) Les modèles de gestion de projet logiciel : ici on met l'accent, en plus de la définition des activités, sur leurs gestion surtout. Par modèle, on veut dire une image abstraite d'un système ou d'une situation. Donc un modèle de gestion est une image qui décrit l'évolution de la construction d'un projet logiciel. On trouve par exemple :
   - Modèle de la cascade
   - Modèle des incréments
   - Modèle de la Spirale :

3 Conclusion

Le GL est la science proposée pour résoudre la crise logicielle, et assurer le développement de logiciels de qualité. Cette discipline doit établir un ensemble d'approches de développement, et de modèle de gestion gérant le processus logiciel. Du côté qualités requise, on peut citer les plus exigées comme suit :

- Fiabilité ;
- Efficacité : le logiciel doit utiliser rationnellement ses ressource : ne doit pas consommer trop de mémoire, ni gaspiller ses cycles d'horloge ;
- Interface appropriée : l'interface doit être facile à utiliser, et à apprendre ;
- Maintenabilité

Cours 2 :

II) Approches de développement et modèles de Gestion de Projet Logiciel

L'objectif de cette partie du cours est de présenter, en grosso-modo (sans trop de détails), les approches de développement et les modèles de gestion de projets logiciels, les plus reconnus dans le monde de génie logiciel. L'accent est, surtout, mis sur l'approche (ou le modèle) de la cascade.

1 Approche de la cascade

1.1 Naissance et idée de cette approche

C'est la première approche proposé en 1969 par Royce (donc connu sous le nom de Modèle de Royce). Cette approche est basée sur les principes suivants :

- Le processus est considéré comme une suite séquentielle d'activité ;
- Chaque activité a une date début et une date fin prédéfinies ;
- Chaque activité une fois terminée, **délivre** (fournit) un **délivrable** (un document) à l'activité qui suit : pour cette raison on parle aussi d'une approche à base de **délivrable**.

1.2 Schémas du modèle : activités

L'approche est schématisée sur la figure 1.

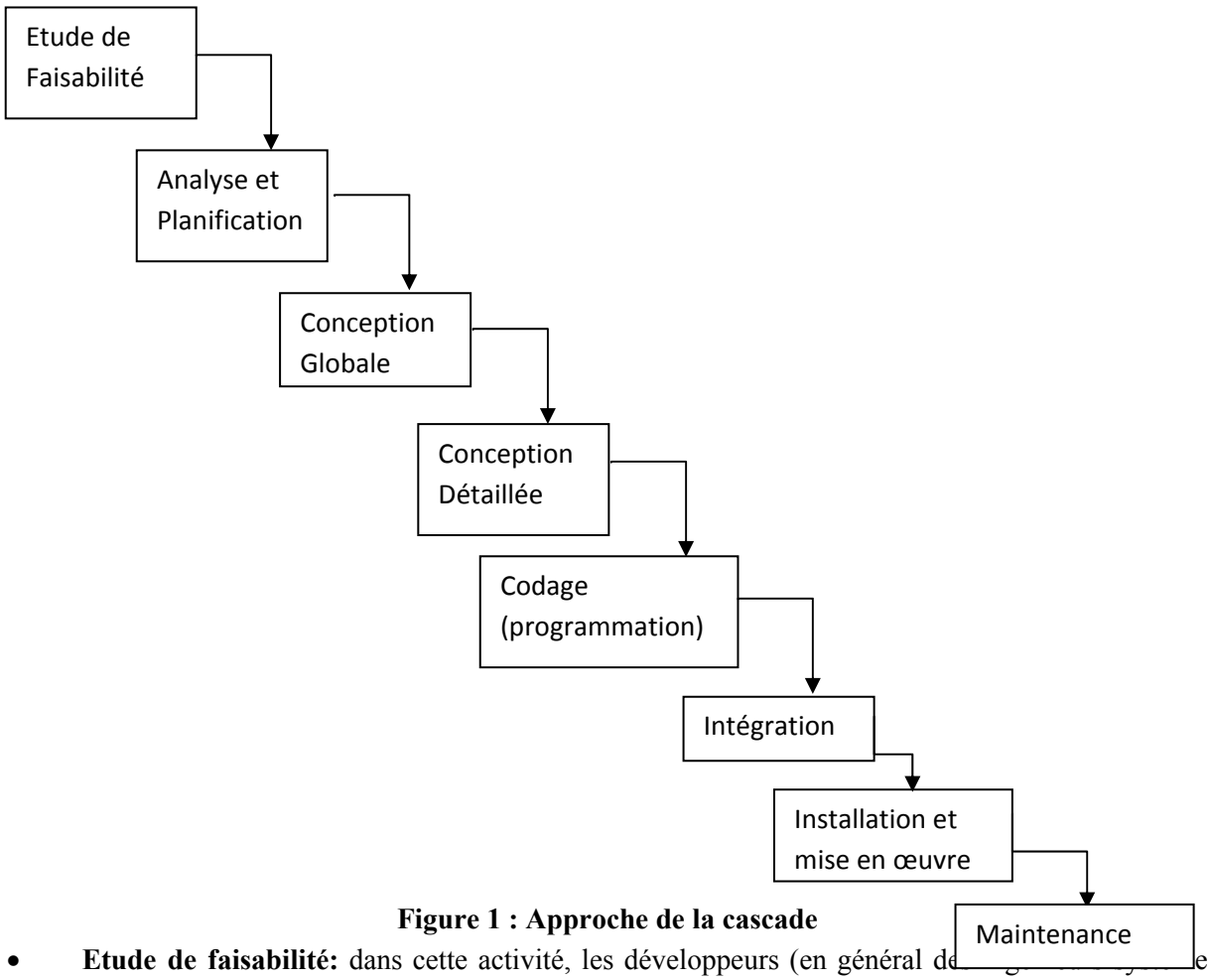

Figure 1 : Approche de la cascade

- **Etude de faisabilité:** dans cette activité, les développeurs (en général de bien expérimentés et disposant de connaissances profondes et des compétences multidisciplinaires)

commencent par se poser des questions comme : est ce que le système est réalisable ou non ? (on donne toujours l'exemple d'un logiciel pour la détection des prochains tremblements de terre qui reste comme un projet trop ambitieux mais non faisable), quelles sont les parties soft de celles hard que le système doit comporter ?, et enfin des décisions seront prises sur l'achat de composants nécessaires, le développement d'autres composants, les approches de développement à suivre, …

- **Analyse et planification**:

Analyse: il s'agit de l'analyse des besoins du client ou utilisateur du futur logiciel.

Objectif de cette activité : de connaitre les objectifs et les buts du client. En général les buts et objectifs sont des énoncés louable (et donc non mesurables : par exemple : joli, rapide, efficace, …), l'analyste doit donc transformer ces buts et objectifs en des besoins mesurables (vitesse de lancement<4 seconde, interface avec la couleur bleue, …). Donc cette activité doit identifier une liste de besoins. Il s'agit donc de répondre à la question **Quoi** ? et non pas à la question **comment** ?.

Personnel requis : les analystes (des développeurs), le client, l'utilisateur, le chef projet logiciel, un maitre d'ouvrage pour rédiger le cahier de charge, …

Outils utilisés : en général dans cette activité, les analystes font appel à des questionnaires, des entretiens, des interviews avec leurs clients et utilisateur pour faire sortir le maximum de besoins et pour se rapprocher de ce que les clients souhaitent avoir dans leur logiciel.

Planification: c'est de planifier le reste du processus. Des gestionnaires travaillent dans cette activité pour préparer le budget, le calendrier, ainsi que les moyens et personnel du processus.

- **Conception globale**:

Objectif de cette activité : définir l'architecture globale du système, et donc identifier les services que le système doit offrir comme réponses aux besoins identifier en analyse. Une liste de services est identifiée à ce niveau. Le système est décomposé en un ensemble de sous-systèmes. Un sous-système sera composé d'un ensemble de modules, et chaque module peut contenir plusieurs unités.

Personnel requis : des concepteurs (des programmeurs connaisseurs d'outils de description de haut niveau, des langages de modélisation, …).

Outils utilisés : langage de modélisation : DFD (diagramme de flux de données), DES (diagrammes entités association, diagrammes de contrôle, UML (Unified Modelling Language), …

- **Conception détaillée**: Pratiquement c'est une phase de raffinement et d'enrichissement des résultats obtenus auparavant. Cette activité doit aboutir à offrir une description assez détaillée du logiciel en construction. Cette description doit être trop proche d'une implémentation, et donc faciliter la tâche de programmation. La description finale peut être sous forme d'un ensemble d'algorithmes détaillés avec des structures de données (dans le cas d'une conception fonctionnelle). Dans cette activité, on utilise souvent des langages de description de programmes comme le langage inspiré de ADA (l'un des langages de programmation offrant une haute lisibilité et clarté des codes).

- **Codage (programmation)**: A ce niveau, il s'agit de choisir un langage de programmation, des environnements de développement, et de commencer le codage (l'implémentation) de la conception détaillée offertes par l'activité précédente.

- **Intégration**: En effet, dans les grands systèmes, le systèmes est composé de plusieurs sous-systèmes qui peuvent être développés, conçus et programmés par différentes équipes et en parallèle. A la fin de la programmation de tous ces sous-systèmes, la tâche d'intégration de tous ces sous-systèmes est importante pour former une seule unité. L'intégration nécessite, surtout, de définir les différentes interfaces entre les différents sous-systèmes réalisés.

- **Installation et mise en œuvre**: L'installation consiste à prendre le système (déjà développé sur la machine de la compagnie de développement), et de le mettre dans la machine du client, et ainsi de le configurer avec cette nouvelle configuration. La tâche d'installation doit rendre le logiciel opérationnel vis-à-vis la configuration matérielle et logicielle du client. Après cette installation, le client commence l'exploitation de son système (donc c'est la mise en œuvre), et de là le client explore les services qu'il souhaite retrouver, et éventuellement des erreurs commencent à être détecter.

- **Maintenance**: elle a pour objectif général de garder le système opérationnel, le plus longtemps possible. Cette activité est une activité permanente. On peut distinguer entre trois types de maintenance :

  - **Corrective** : dont l'objectif est de corriger les erreurs découvertes ;
  - **Adaptative** : dont l'objectif est d'adapter le système aux nouvelles technologies matérielles et logicielles ;
  - **Perfective** : dont l'objectif est d'améliorer le système est d'augmenter ses performance.

1.3 Critiques de cette approche

L'approche de la cascade est la plus ancienne. Elle a reçu certaines critiques et plusieurs améliorations et remédiassions ont été proposées. Parmi les critiques :

- Un modèle trop idéal et ne reflète pas la réalité : les activités nettement séparées, sans retour en arrière, sans chevauchement. En réalité, ces activités ne sont pas assez séparées ;
- La gestion des délivrables: ces documents exigent des lectures, des corrections, des mises à jour. Ces opérations consomment du temps, du personnel et de l'argent aussi.
- Apparition des délivrables artificiels : quelques fois les développeurs risque de produire des délivrables artificiels pour seulement présenter de rapports à leurs gestionnaires, sans que ces délivrables reflètent le développement réel.

Pour répondre à ces critiques et à d'autres, les chercheurs ont proposé d'autres approches de développement ainsi que d'autres modèles de gestion de projets logiciels.

1.4 Représentation en V (représentation améliorée de la cascade)

1.4.1 Principe :

Une autre représentation de la cascade est le modèle en V. Ce modèle mets l'accent sur certains aspects non présents dans la représentation de la cascade :

- Activité de test : avec différente variante ;
- Relation entre les différentes activités

Dans cette représentation, on distingue entre deux types d'activités :

1) Les activités constructrices : dont l'objectif est de développer le système. On parle de construction du système aussi : Faisabilité, Analyse, Conception, et codage
2) Les activités destructrices : dont l'objectif est de trouver les erreurs commises durant les activités de construction. Donc, c'est une forme de destructions du travail construits. En effet, il s'agit d'une suite d'activités de test qui réussissent quand elles identifient des erreurs dans les produits des premières activités : Test unitaire, test d'intégration, test d'acceptation, et enfin le test système

La représentation en V montre explicitement les relations entre ces différents types d'activités: les premières activités de construction préparent les dernières activités de test. Par exemple, ce qu'on doit faire dans un test système doit être préparé et planifiée durant la phase d'analyse elle-même.

1.4.2 Schéma du V

La figure 2 montre le modèle en V avec ses différentes activités

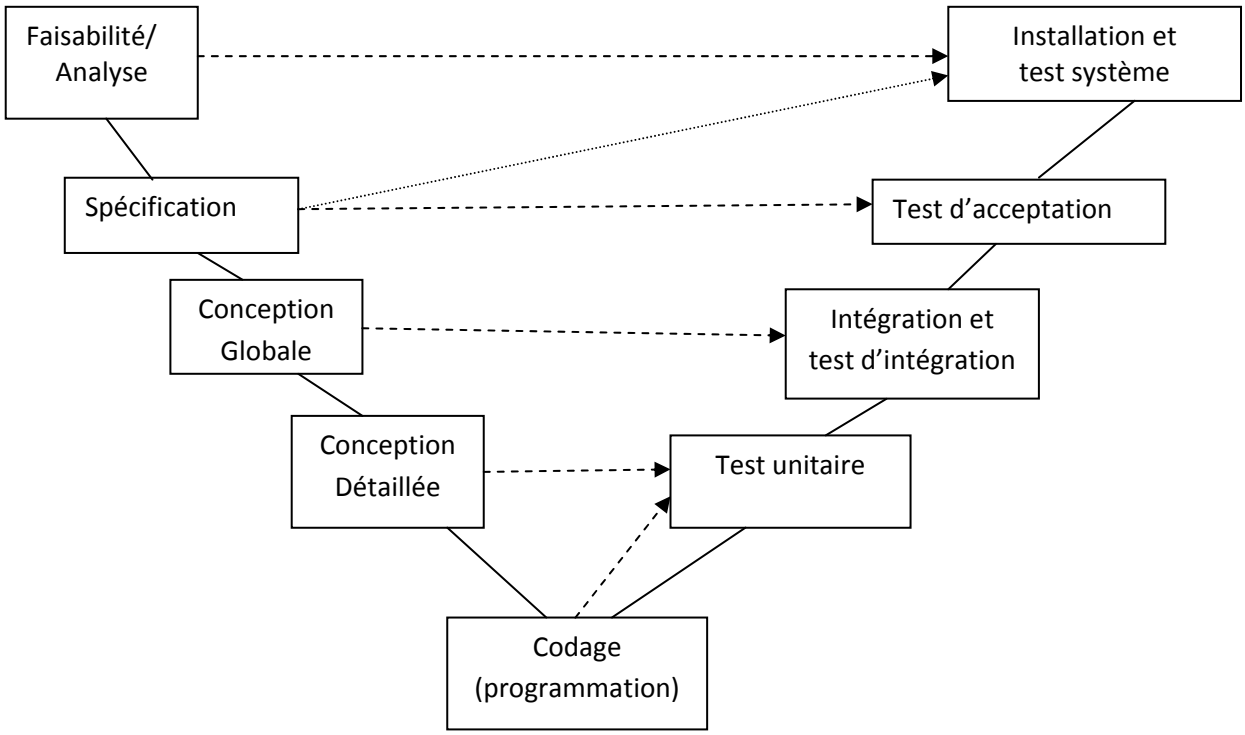

Figure 2 : Représentation en V

Dans ce schéma, certaines activités sont incluses :

1) L'activité de spécification : dans cette activité, les analystes ou **spécifieurs** prépare une spécification des besoins. Une spécification doit être une description assez concise, précise, sans ambigüité. Une telle spécification peut être même formelle (décrite dans des langages mathématique et logique si le système le nécessite). Une spécification diffère d'un cahier de charge en termes de précision.

2) Le test unitaire : cette activité doit permettre de tester les unités du logiciel de façon séparée, unité par unité. Elle est liée aux activités de codage et de conception détaillée.

3) Le test d'intégration : cette activité s'intéresse à tester les unités intégrées et leurs interfaces de communication. Elle est liée à la conception globale du système.

4) Le test d'acceptation : il est réalisé à la présence du client chez la compagnie du développement et sur sa configuration matérielle et logicielle.

5) Le test système: il est réalisé à la présence chez le client chez sur la configuration matérielle et logicielle cible.

Remarque : dans cette représentation, les arêtes décrivent l'ordre du déroulement d'activités, et les arcs discrétisés décrivent les relations de préparation et de dépendance entre activité de développement de test.