LE GENIE LOGICIEL

Software Engineering

I. Introduction

1) Généralités

Historique

L'origine de la discipline est la réunion du comité scientifique de l'Organisation du Traité de l'Atlantique Nord (à Garmich, RFA 1968) avec quelques-uns des plus grands spécialistes mondiaux de la programmation . Cette réunion avait comme objectif d'étudier les causes de ce que l'on commençait à appeler la crise du logiciel.

Les symptômes de la crise : plusieurs problèmes ont été soulevés lors de cette réunion :

- Problème d'Adéquation : les systèmes informatisés ne correspondent pas souvent aux besoins des utilisateurs.

- Problème de Fiabilité : les systèmes tombent souvent en panne.

- Problème de Modifiabilité : la maintenance est souvent complexe et coûteuse. Il était difficile d'adapter les logiciels pour prendre en considération l'évolution des environnements et des besoins des utilisateurs.

-Coût et délai : budgets dépassés, projets en retard. Le coût du logiciel est devenu supérieur à celui du matériel : 50%-50% en 1965, 70%-30% en 1975, 80%-20% en 1985. Les coûts résultent généralement des frais de maintenance qui découlent de la qualité déficiente du logiciel au moment de sa livraison ou après son évolution.

- Interface utilisateur inappropriée.

Enfin les causes de ces problèmes sont nombreuses et parmi les plus importantes : le caractère rudimentaire des méthodes et outils utilisés dans le développement de logiciels.

Moralité

Cette conférence a montré qu'il est necessaire de faire de la production de logiciels une discipline d'ingénieur à part entière, avec une solide base scientifique (comme dans les autres disciplines telles que le génie civil, le génie chimique etc...). D'où le titre : génie logiciel.

Définition du Génie logiciel (voir Annexe1 pour quelques définitions générales)

Ensemble de méthodes, techniques et outils nécessaires à la production de logiciels de qualité .

⇒ Plus précisément, le génie logiciel vise à satisfaire:

- **Les utilisateurs** : en produisant des logiciels adaptés aux besoins.
- **Les gestionnaires** : en réduisant le coût de production et de maintenance.
- **Les chefs de projet** : en rendant les logiciels productifs dans un délai raisonnable.

Comment y parvenir: plusieurs actions peuvent contribuer à la production de logiciels de qualité :

- Améliorer la démarche de développement : Appliquer et faites appliquer les méthodes d'analyse, de conception (SADT, SART, MERISE, OOA...), dans le développement de logiciels.
- Améliorer les langages : On peut programmer mal dans un bon langage , et assez bien dans un mauvais, Mais il est plus facile de bien programmer avec un bon langage. Il est important de connaître les nouveaux concepts des langages modernes (modularité, abstraction, généricité, héritage..) même si l'on ne peut pas toujours les employer en pratique.
- Utiliser des outils d'aide au développement :

-Les environnements de programmation : éditeur syntaxique, outils de tests,...

-Les ateliers de génie logiciel : outils d'aide au spécification, gestionnaire de configuration, gestionnaire de projets,.....

- Décomposer le développement en plusieurs étapes : Suivre le cycle de vie du logiciel. C'est ce que nous allons étudier dans la suite en précisant les étapes et les différentes approches de développement.

2) LE CYCLE DE VIE DU LOGICIEL : Software life cycle or Software process

Qu' appelle t'on logiciel ?

L'organisation internationale de normalisation (ISO) a défini en 1981 le logiciel (software) comme une création intellectuelle rassemblant :des programmes permettant de faire fonctionner un ordinateur et de l'utiliser pour résoudre des problèmes, et la documentation servant à concevoir, réaliser, utiliser et modifier ces programmes.

Cycle de vie du logiciel

Ensemble des étapes qui composent le processus de développement et d'utilisation du logiciel.

Modèle du cycle de vie

Modélisation conventionnelle de la succession d'étapes qui préside à la mise en oeuvre d'un produit logiciel. Il s'agit de décrire et de représenter les étapes du cycle de vie et leurs relations. Ainsi plusieurs modèles ont été proposés : la cascade (Waterfall model), cycle en V, modèle en spiral,.....

Pourquoi de tels modèles? (Voir Annexe2 pour la définition de la modélisation et de la notion de modèle)

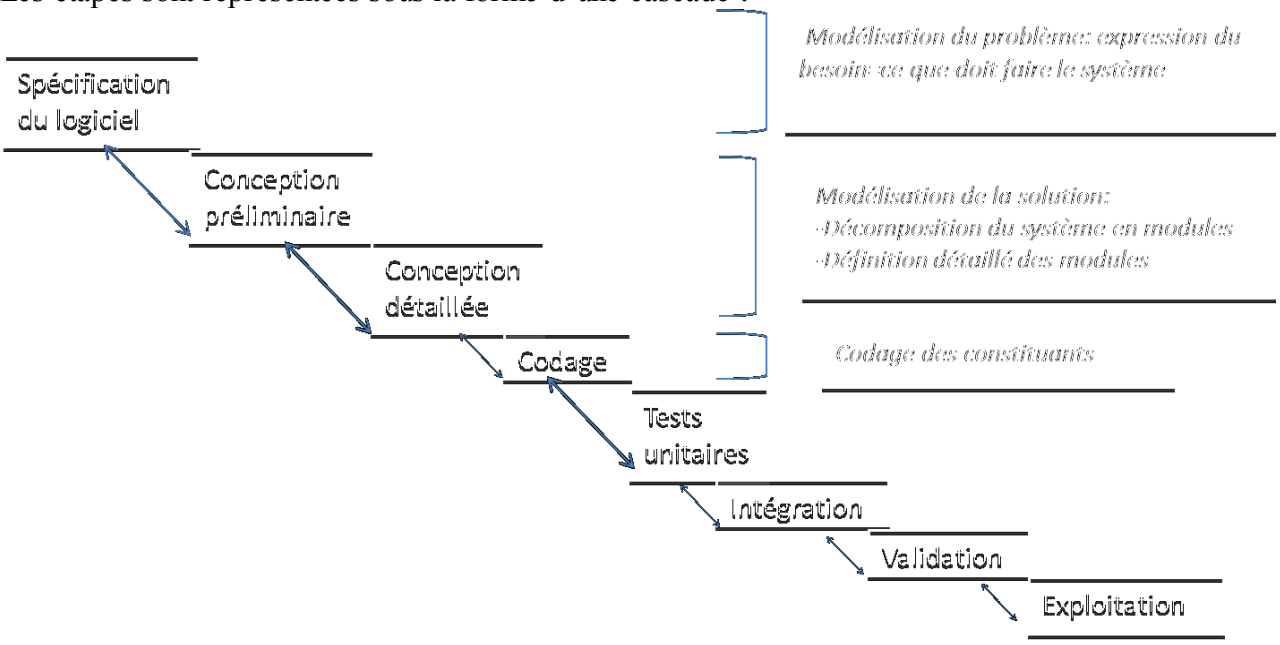
Un modèle du cycle de vie (cascade, ou en V ou en spirale ou…) est une référence pour les personnes impliquées dans le développement d'un logiciel. En effet, un modèle du cycle de vie ::

- Permet de représenter le processus de développement de manière graphique et physique,
- Donne une structure autour de laquelle les activités d'assurance qualité peuvent être construites de manière utile et disciplinée,
- Fournit un repère pour les acteurs d'un projet logiciel

Nous présentons ci-dessous les deux modèles les plus anciens (Cascade et en V), puis la définition des étapes communes à tous les modèles de développement. Après, nous reviendrons sur les autres modèles : spirale, par prototypage, incrémental, Agile.

Modèle de la cascade

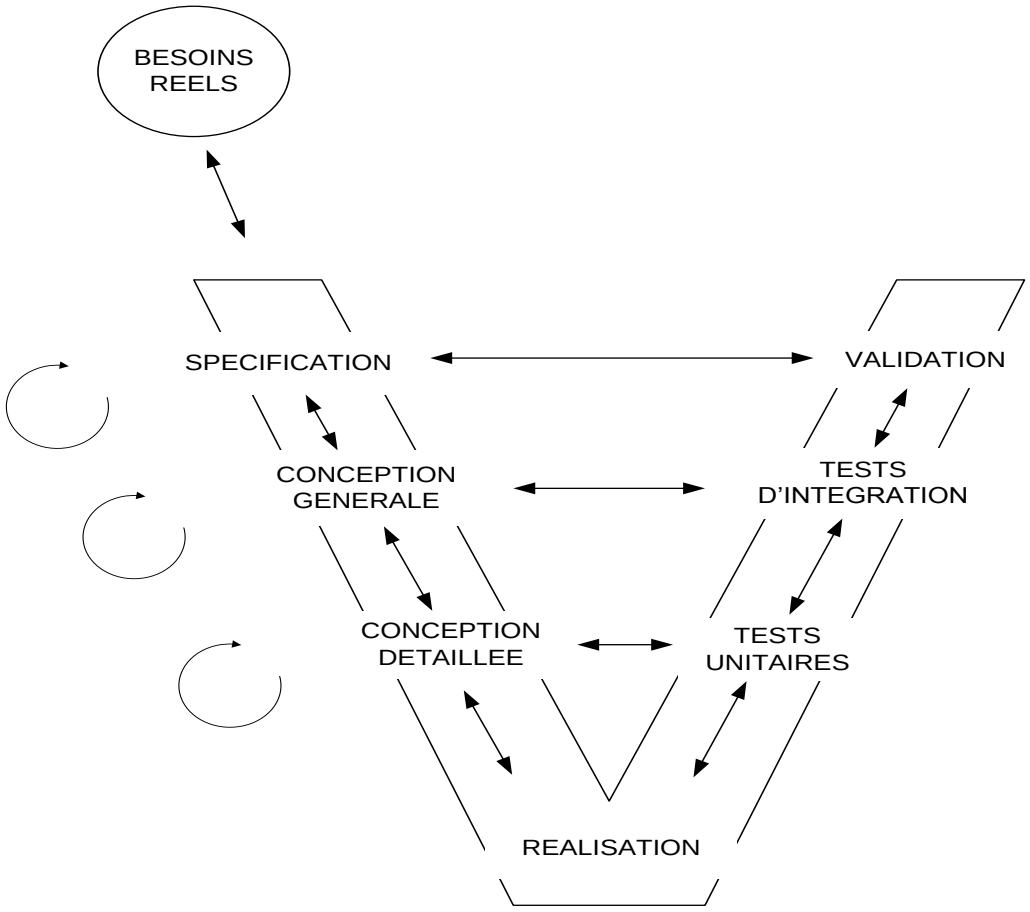
Les étapes sont représentées sous la forme d'une cascade :



Les principales caractéristiques du modèle de la cascade sont:
-Chaque phase ne peut démarrer que si la précédente est finie.
-A partir d'une phase, on ne peut revenir qu'à celle qui la précède.
-Facile à adopter et à comprendre
-Idéal pour le suivi de projets et le contrôle qualité

Ce modèle repose sur une vue idéaliste du processus de développement et pésente par consequent un certain nombre d'Inconvénients :
-Mal adapté à des systèmes complexes ou nouveaux
-Suppose une définition complète de tous les besoins, ce qui est généralement difficile,
-Délai assez long pour disposer d'une version du système
-Manque de souplesse et de réactivité

Modèle en V
Il s'agit d'une amélioration du modèle de la cascade consistant à anticiper et préparer dans les étapes descendantes les « attendus » des futures étapes montantes (tests et validation). Ainsi, à chaque phase de développement correspond une phase de contôle de conformité.



Bien qu'il apporte plus que le modèle de la cascade, le modèle en V présente aussi une vue idéaliste du cycle de vie de développement et hérite certains de ses défauts. C'est pourquoi, d'autres modèles ont été proposés, comme nous le verrons plus loin, après la présentation de la définition de chaque étape de développement.

II. Les étapes du cycle de vie

1. L'ETUDE PREALABLE

1.1 Définition

BUTS :

- Préciser l'objet du système : Comprendre les besoins et les enjeux,
- Dialoguer avec les différents types d'utilisateurs
- Esquisser des solutions
- Estimer les coûts et délais
- Préparer un dossier en vue de décision

Peut se faire avec l'assistance d'une société de conseil ou d'ingénierie

RESULTATS :

- Décision de développement.
- 1ère version du contrat de développement, 1ère estimation du coût de développement.
- Cahier des charges, Mode de fonctionnement grossier du système.
- Plan général du projet.

MOYENS :

- Plan d'entretien avec le client (techniques d'entretien).
- Certaines méthodes MERISE, AXIAL....
- Revue pour valider et approuver les propositions exhibées.

1.2 Cahier des charges

- Résultat des études préalables menées, directement ou indirectement par le maître d'ouvrage.
- Bases de l'appel d'offre ou de la consultation.
- Doit donner tous les éléments nécessaires pour permettre au maître d'oeuvre d'élaborer et chiffrer des solutions.
- Doit rester strictement au niveau de l'expression du besoin et des exigences fonctionnelles, sans imposer le choix de conception :
- permettre aux soumissionnaires de faire dans leur contexte la meilleure offre
- bénéficier de leurs éventuelles solutions innovantes.

1.3. Organisation du cahier des charges

- Les clauses techniques

-> Fonctionnalités du logiciel à réaliser : exprime le besoin de l'utilisateur en décrivant de manière complète et non ambiguë les fonctionnalités du logiciel à réaliser.

-> Contraintes à prendre en compte.

- Les clauses de qualité

Obligations du fournisseur en assurance & contrôle qualité pendant le déroulement du projet.

- Les clauses générales

Dispositions : juridiques, financières, commerciales auxquelles la proposition du fournisseur devra se conformer

2. LA SPECIFICATION

Requirements phase, analysis phase

BUTS : **Définir le "quoi"**

-> Obtenir une description précise et sans ambiguïté du système logiciel - Support pour la validation

-> Définir la portée et les objectifs du projet de manière précise.

- Pour le Produit logiciel, définir: les performances requises, les fonctions du logiciel, les objets du monde réel manipulés, les interfaces du logiciel avec son environnement, lesentrées et sorties.

- Pour le Projet, définir: les ressources, les contraintes de réalisation.

RESULTATS :
Manuel d'utilisation , Document de spécification (spécification détaillée des besoins).
Plan détaillé du reste du projet, Raffinage de l'estimation des coûts.

MOYENS :
Méthodes, outils, langages de spécification. *Exemple : SADT, SREM, OOA.*.
Revues de spécification. Techniques de planification.

Qualités d'une Spécification

1. Fidélité : reflète correctement le besoin
2. Complétude : couvre complètement et exclusivement l'ensemble des besoins
3. Cohérence (non contradiction) : les compromis doivent être mis en évidence
4. Lisibilité : sert de vecteur de communication
5. Indépendance : par rapport aux choix d'implémentation

3. LA CONCEPTION GENERALE

Preliminary design or architectural design
BUTS : Décrire l'architecture du logiciel
- Obtenir une décomposition du système en un ensemble de modules
- Obtenir une description précise du rôle de chaque module.
- Vérifier la faisabilité et la traceabilité vers les spécification :trace (lien) entre les fonctions du système et les modules logiciels.

RESULTATS
-> Document de conception : description de la structure modulaire du système - Définition des modules et des interfaces. Il faut conserver une trace entre les fonctions du système et le ou les modules introduits.

-> Mise à jour du manuel d'utilisation
MOYENS
-> Méthodes, langages, outils de conception : HIPO, STRUCTURED DESIGN, JACKSON, HOOD.
-> Revues de conception.

4. LA CONCEPTION DETAILLEE

Detailed design
BUTS :
Obtenir une description détaillée des Modules, des traitements et des structures de données permettant un codage immédiat.
RESULTATS :
Dossier de conception détaillée. Planification des tests unitaires.
Mise à jour : DS, MU
MOYENS :
"Pseudo-codes", outils (PDL).
Langages de programmation ?
Revues de conception.
NOTE : Conception Detaillée = Programmation Abstraite.

5. LE CODAGE
Coding

BUTS :
Obtenir les programmes : traduire la conception détaillée dans un langage de programmation en respectant les standards définis.

RESULTATS :
Programmes.
Documentation technique.

MOYENS :
Langages de programmation. Editeurs syntaxiques,....
Revues de codes ("inspections").

6. LES TESTS UNITAIRE Unit test phase

BUTS :
Tester chaque module séparément : exécuter le corps du module et comparer son comportement à un comportement de référence. Il faut élaborer un jeu de données de tests pour chaque module. (attention le test n'est pas une preuve).

RESULTATS :
Résultats des tests (+ rapport d'erreur).

MOYENS : Analyseurs statiques, tests dynamiques, outils de test.

7. INTEGRATION Integration phase

BUTS : Recomposer le logiciel en modules à partir de ses composants.
Vérification des fonctionnalités et des performances du système complet.
Vérification du respect des normes de programmation.
Vérification de la documentation.

RESULTATS :
Système logiciel intégré.
Documentations internes et externes.

MOYENS :
Utilisation de jeux de tests.
Outils d'évaluation de performances.

8. VALIDATION

BUT : Tester le logiciel dans des conditions représentatives de son utilisation opérationnelle.
Valider le logiciel par rapport à la spécification des besoins

RESULTATS :
Documents de validation. PV de qualification, rapport qualité-fiabilité
Programme certifié et installé.

MOYENS :
Utilisation de jeux de tests. Contrôle qualité

9. EXPLOITATION ET MAINTENANCE

Cette phase ne fait pas partie du projet proprement dit, mais à l'issue d'un développement, il s'avère toujours que des modifications doivent être apportées au produit :
-Correction d'erreurs décelées au cours de l'exploitation,
-Modification du logiciel pour satisfaire de nouveaux besoins,
-Optimisation de certaines fonctions.

Annexe 1 : Complément de définitions (certaines sont issues sans les exemples de J. Girerd)

- **Art de l'ingénieur**: Créer des produits -> Marché

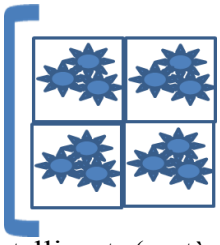
Génie: méthodes, techniques et outils pour créer des produits

Exemple : Génie électrique, Génie logiciel, Génie des télécommunications

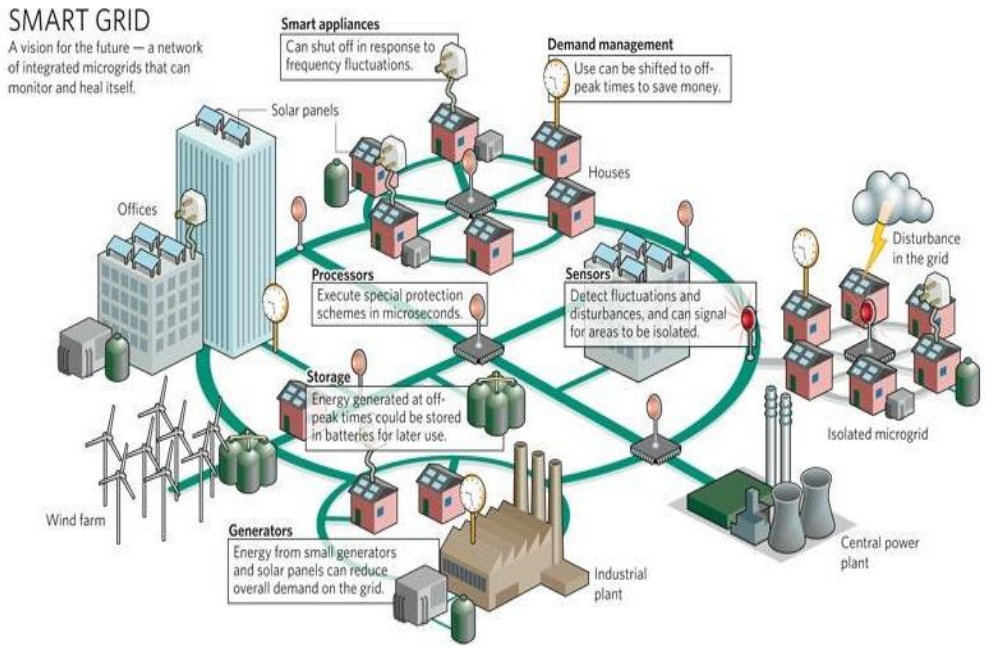
- **Ingénierie:** méthodes, techniques et outils pour créer des systèmes par mise en œuvre de divers génies et assemblage de produits

Exemple : système embarqué dans un véhicule routier

- **Intégration**: méthodes, techniques et outils pour créer des systèmes par assemblage de systèmes



Exemple : un réseau électrique intelligent (système) composé de plusieurs systèmes ou micro-réseaux (voir ci-dessous) . Il s'agit d'un système de systèmes (System of systems SOS) qui permet de gérer la production et la distribution de l'énergie à l'echelle d'une ville par exemple.



https://www.scenarios2020.com/2011/07/la-smart-grid-r%C3%A9seau-intelligent-de-transport-de-l%C3%A9lectricit%C3%A9.html

Annexe 2 : notion de modèle

La Modélisation est un processus d'abstraction qui consiste à identifier les caractéristiques intéressantes d'une entité, en vue d'une utilisation précise. Il s'agit d'une activité commune à plusieurs disciplines scientifiques et domaines d'ingénierie : physique, électronique, mathématiques, etc

Un Modèle est le résultat de l'activité de modélisation. C'est une abstraction ou représentation simplifiée d'une entité (ou d'un système) permettant de la comprendre, d'en prédire le comportement, de l'étudier. Il décrit donc, l'ensemble des caractéristiques essentielles d'une entité, retenues par un observateur, pour un objectif donné.
D'un point de vue pratique, un modèle réduit la réalité dans le but de disposer d'une représentation du système exploitable par des moyens mathématiques et/ou informatiques

Exemples :
-Modèle météorologique: à partir de données d'observation (satellite, mesures, …), permet de prévoir les conditions climatiques pour les jours à venir.
-Modèle conceptuel de données d'un système: représente de façon structurée les données d'un système et leurs relations