Chapitre 1. Évolution de la Conception des SI

- **Objectifs du cours** :
  - Se rappeler des différents cycles de vies d'un SI,
  - Comprendre le rôle et importance d'une méthode de conception,
  - Avoir un aperçu des différentes familles de méthodes de conception existantes.

1. Cycle de développement d'un SI

Se compose de 3 grandes phases :

- Phase d'**Analyse des besoins** (25% du temps).
- Phase de **Conception** et de **Réalisation** (55% du temps)
- Phase de **Mise en place** (mise en œuvre) (20% du temps)

A. Analyse : Étude d'un objet ou d'une situation pour comprendre le fonctionnement dans un but d'amélioration. Elle définit les fonctions attendues du SI

Avant de se lancer sur un projet, on doit faire une Analyse comme suit :

- Identification des anomalies (erreurs) et problèmes à résoudre,
- Identification des besoins des utilisateurs,
- Identification des objectifs à atteindre,
- Identification des acteurs (intervenants),
- Étude de l'existant
  - Flux d'information
  - Étude de tous les documents (factures, formulaires, notes…)
  - Étude des solutions existantes
  - Critiques de l'existant
- Étude de faisabilité
  - Technique : est-ce que c'est faisable ?
  - Économique : avantages à tirer + évaluation des coûts (développement, fonctionnement, solutions techniques…)
- Planning : Calendrier de réalisation

  - Phase d'analyse —concerne→ **Collecte d'informations** via
    - Entretiens
    - Présentations
    - Questionnaires
    - Observations
    - Rapports + documents…

B. Conception et Réalisation

B1. Conception : création d'un objet ou d'un système. C'est une action qui donne naissance à quelque chose qui n'existait pas.

Conception = Solution proposée (Solution informatique)

La conception est basée sur les besoins des utilisateurs, elle fournit une description fonctionnelle du système qui consiste à construire une architecture capable de répondre à ces besoins.

Lors de la conception d'une solution, on doit :

- Modéliser avec des diagrammes et schémas le fonctionnement du nouveau système
- Structurer les données (format prédéfini, codification, contrôles…)
- Définir les postes de travail (Rôles, qui fait quoi ?)
- Organiser les traitements (procédures, fonctions, calculs, algorithmes…)
- Choix des techniques : Matériels (machines, réseaux, architecture Client-Server 2 tiers, Client-Server 3 tiers…), Langages de programmation (Java, Python, C, PHP…), Systèmes de gestion de base de données SGBD (MySQL, Oracle, SQL serveur…)
- Dictionnaire des données.
- Base de données (BDD).
-
    - Phase de Conception $\xrightarrow{\text{concerne}}$ **Stockage** et **Traitement** des **informations**

B2. Réalisation : consiste à développer le logiciel (solution proposée).

Lors du développement de la solution, on s'intéresse à :

- Écriture des programmes (Code source)
- Implémentation des fonctionnalités
- Développement des interfaces graphiques (IHM)
- Connexion entre le code source et la BDD.
- Réalisation des versions du logiciel.
- Tests et jeux d'essai (Validation + correction des bugs)

    - Phase de Réalisation $\xrightarrow{\text{concerne}}$ **Traitement** et **Diffusion** des **informations**

C. Mise en place (Mise en œuvre) : il s'agit du déploiement de la solution réalisée (hébergement, installation, configuration, mise en service).

Lors de la mise en œuvre de la solution, on doit assurer :

- ***Basculement*** : de l'ancien système vers le nouveau (mise en place du nouveau système et formation des utilisateurs).

- ***Maintenance*** : suite aux erreurs ou à une évolution des besoins.

Contrat de maintenance ⟶ ***Licence***

Remarque : il existe différents modèles des cycles de vie des SI :

- ***Séquentiel*** *(Cascade, V, W…)*
- ***Itératif ou Évolutif*** *(Spirale, RAD…)*
- ***Objet*** *(Cycle en Y)*

2. Modélisation

C'est la présentation d'un phénomène complexe sous forme de **modèles**.

Pourquoi modéliser =>
- Comprendre ce qui se passe
- Simplifier
- Simuler des comportements + communiquer des descriptions
- Abstraire la réalité pour mieux comprendre le système à réaliser
- Reproductivité (Reproduire + Réutilisation)

a. Modèle

- Un modèle est une représentation simplifiée d'une réalité sur laquelle on veut se renseigner. (Exemple : un plan, une carte, un schéma…).
- Un modèle s'exprime avec un ensemble de concepts dotés de règles d'utilisation et de représentation (souvent graphique).

b. Méthode

- La méthode représente une démarche et est un ensemble de modèles permettant de développer un nouveau système.

Méthode = Modèles + Démarche + Langages + Outils

- Les méthodes sont généralement créées par des organisations de normalisation, des universités, des grandes entreprises, etc.

b.1. Intérêts d'une méthode de conception

Les méthodes de conception ont un rôle primordial et très important dans la gestion d'un projet. Parmi les intérêts de ces méthodes, on peut citer :

- Permet d'éviter les faiblesses du langage naturel (interprétations spécifiques de chacun, ambiguïté des propos…).

- Utilisation de modèles standards et déjà définis (connus) qui facilitent le travail en équipes.
- Modélisation des besoins du client (à partir d'un énoncé informel tel exprimé par le client → système ou logiciel informatique qui répond aux besoins et qui satisfait le client.
- Assurer un haut niveau de qualité aux systèmes conçus en respectant bien les concepts et guides de la méthode.

b.2. Familles ou Catégories de méthodes de Conception

Il existe plusieurs catégories (familles) de méthodes, à savoir :

- ***Méthodes analytiques ou cartésiennes*** (fonctionnelles) : comme Corig en 1960, RSA en 1977 => Apparition des Langages de programmation.
- ***Méthodes systémiques*** : comme E/R Chen en 1975, MERISE en 1978, Axial en 1986, Remora en 1988 : un système est composé de 3 sous-systèmes (SP, SI et SO) => SGBD, SQL, Conceptualisation et Normalisation.
- ***Méthodes à objets*** : comme OMT (POO) en 1991, Booch en 1994, UP (UML) en 1999 : le système est organisé comme une collection d'objets => Technologies Internet, Nouvelles architectures Client-Server 2/3 ou 3/3, BDD objet, etc.
- ***Méthodes agiles*** : comme SCRUM, XP, Crystal, RAD, etc. => Le client au centre du développement.

Comme il existe plusieurs et différentes méthodes de conception, et l'application d'une de ces méthodes est un processus difficile. Des outils automatiques sont apparus pour faciliter l'application d'une ou plusieurs méthodes. Il s'agit des AGLs.

3. AGL (Atelier de Génie Logiciel)

a. Définition : un AGL (Atelier de Génie Logiciel) ou CASE (Computer-Aided Software Engineering) est un *ensemble* intégré d'*outils* qui permet aux développeurs de logiciel de **documenter** et **modéliser** un SI dès la spécification initiale des besoins jusqu'au projet et son implantation, en passant par les **tests** de *cohérence*, de *complétude* et *conformité* aux spécifications proposées.

b. Objectifs d'un AGL

- Aider l'utilisateur dans l'application de la méthode de développement,
- Faciliter l'utilisation des techniques de modélisations,
- Faciliter la production de documents,
- Accélérer la production des systèmes,

- Organiser le travail d'équipes dans un projet (Calendrier, répartition des tâches…),
- Intégrer les différents produits ou résultats issus des différentes étapes du cycle de vie (analyse, conception, réalisation, tests…),
- - Aide à l'édition des programmes dans différents langages de programmation, avec compilation et génération du code source,
- Génération et création rapide des IHMs (Interfaces Homme Machine),
- Aide aux tests et suivi de correction (débogage),
- Contrôler la cohérence et la complétude des systèmes produits,
- Gestion des versions successives d'un même programme,
- Reverse Engineering,
- …

c. Types d'AGLs

On distingue essentiellement deux types d'AGL selon la nature des outils intégrés :

1. *Les environnements de conception (**upper-case**):* ces ateliers s'intéressent plus particulièrement aux phases d'analyse et de conception du processus logiciel. Ils sont basés sur une méthode de conception. Ils intègrent en général des outils pour l'édition de diagrammes, des dictionnaires de données, édition de rapports, des générateurs de (squelettes de) code, des outils pour le prototypage, etc.

2. *Les environnements de développement (**lower-case**):* ces ateliers s'intéressent plus particulièrement aux phases d'implémentation et de test du processus logiciel. Ils intègrent en général des éditeurs, des générateurs d'interfaces homme/machine, des SGBD, des compilateurs, optimiseurs, debuggers, ...

d. Exemples d'AGLs

- StarUML
- ArgoUML
- Objecteering
- Entreprise Architect
- Papyrus