I.2.4.2 Donnée multidimensionnelle

Il existe principalement deux types de base de modèle dimensionnel :

- schéma en étoile
- schéma en flocons de neige

a. Schéma en étoile

Le schéma en étoile est une façon de représenter un modèle dimensionnel au sein d'une base de données relationnelle. Un schéma en étoile (Figure 1.9) se constitue d'une table de faits associée à un ensemble de tables de dimensions. Ces dernières sont toujours directement reliées à la table de faits.

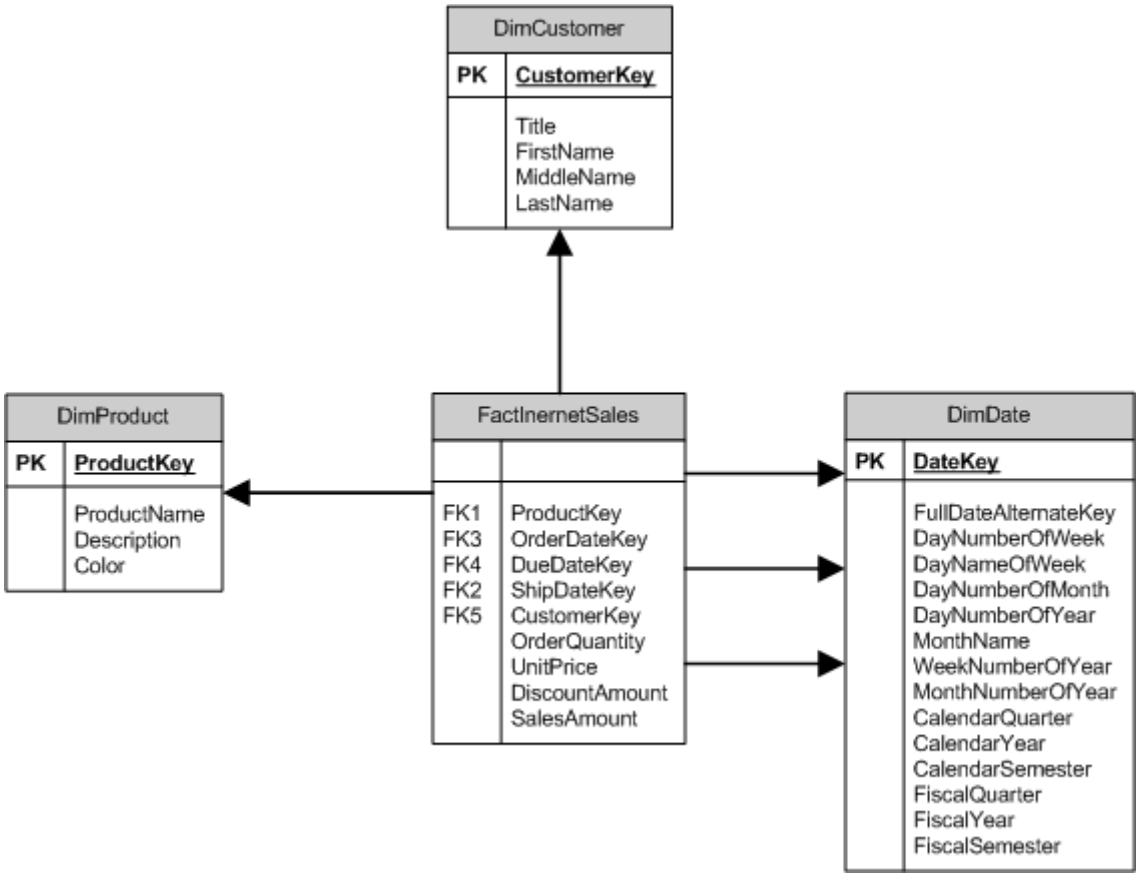

Figure 1.9 : Schéma en étoile des dimensions

b. Schéma en flocons de neige

Un schéma en flocon est une représentation normalisée (3NF) d'une unique table de dimension. Cette table, qui peut parfois contenir un grand nombre d'attributs, est scindée en plusieurs entités. Il est basé du schéma en étoile mais la dimension est repartie en plusieurs entités. Les figures suivantes montrent des exemples d'un schéma en étoile et en flocon :

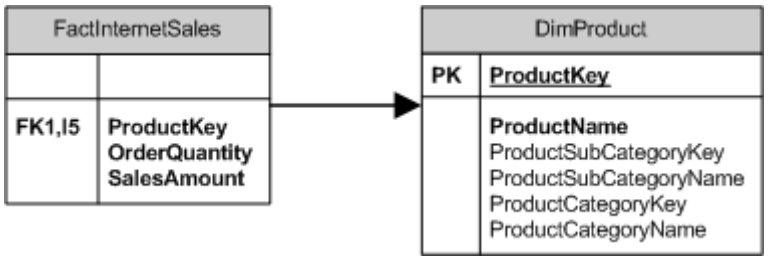

Figure 1.10 : Dimension Produit modélisée sous la forme d'un schéma en étoile

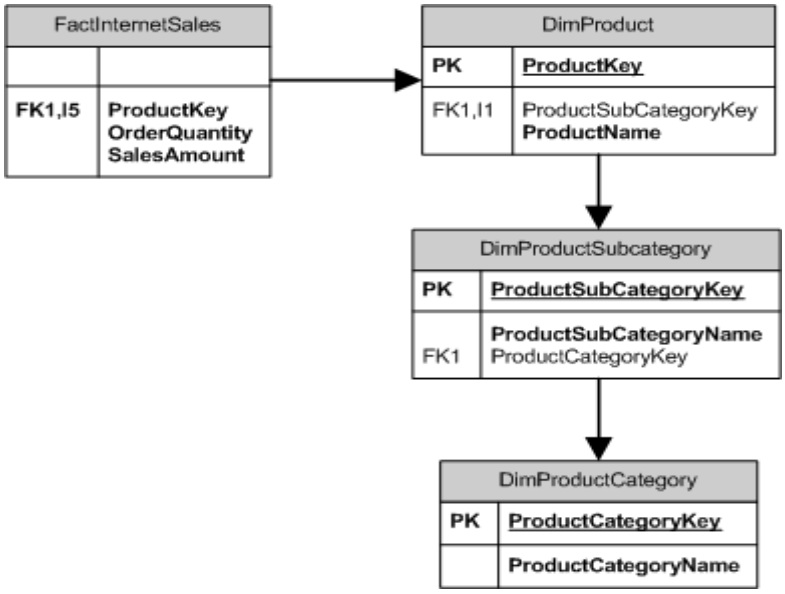

Figure 1.11 : Dimension Produit modélisée sous la forme d'un schéma en flocon

I.2.4.3 Processus décisionnel

Toute décision est une prise de risque, le rôle du BI est de limiter au maximum le risque et d'apporter de valeur ajoutée à la décision.
Les décisions dans l'organisation sont liées aux activités qui s'y déroulent et en général on ne peut pas prendre de décision si on n'a pas encore défini les objectifs.

a. *Type de décision*

Il y a trois niveaux de management dans l'organisation :

- régulation (contrôle opérationnel) : activités concernant principalement des rythmes inférieurs à un mois et conduit à des décisions de portée limitée (champ local et limité)
- pilotage (Planification et le contrôle managériaux) : activités conduisant à des décisions dont les conséquences sont à moyen terme
- planification Stratégique : activités engendrant des décisions majeures dont les conséquences sont à long terme. La décision est de portée globale (champ global)

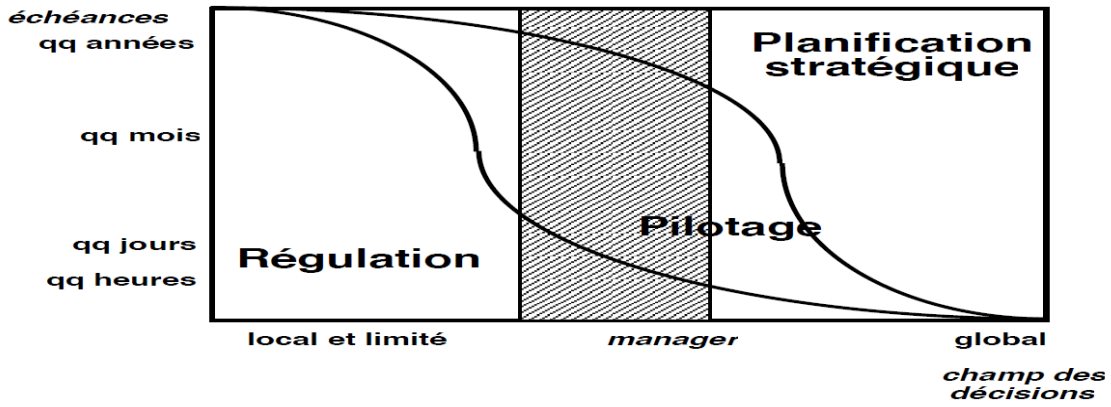

Figure 1.12 : Champs des décisions et Management de l'organisation

De façon générale, pour chaque niveau managérial, la répartition des types de décision concernés est définie par la figure suivante :

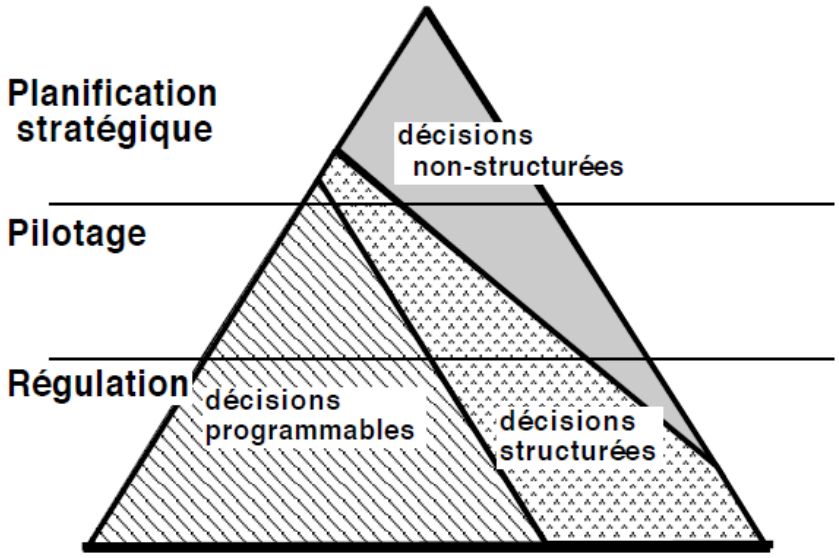

Figure 1.13 : Type de décision et Management de l'organisation

Cette configuration mène à proposer deux types de décisions :

- décisions programmables : décisions répétitives et routinières, et une procédure a été définie pour les effectuer, évitant ainsi d'avoir à les reconsidérer chaque fois qu'elles se présentent
- décisions non programmables : il n'a pas été possible de définir une procédure spécifique pour les effectuer; soit du fait qu'elles sont nouvelles, non structurées, inhabituelles, …

b. Choix de décision rationnelle

Définition 1.7 :

Choix rationnel : une décision cohérente et générant de la valeur dans la limite des contraintes données.

Pour arriver à la sélection d'une décision, on devrait passer par deux étapes (modèle IDC de la figure 1.14) :

- phase d'intelligence (investigation) :
  - processus de formulation du problème décisionnel
  - confrontation entre situation perçue et situation voulue : perception de dissonance
  - définition de valeurs, d'objectifs, de frontières, d'actions (solutions) possibles
- phase de modélisation (conception) :
  - élaboration de modèle, d'actions possibles, de plans d'action intentionnels, de stratégies
  - possibles permettant la résolution du problème
  - décrire/prévoir l'état du système si on lui applique une action possible
- phase de choix (sélection) :
  - évaluation, comparaison, classement des actions possibles
  - choix d'une action parmi ces actions possibles
  - si aucune action n'est satisfaisante, reconsidérer les phases antérieures

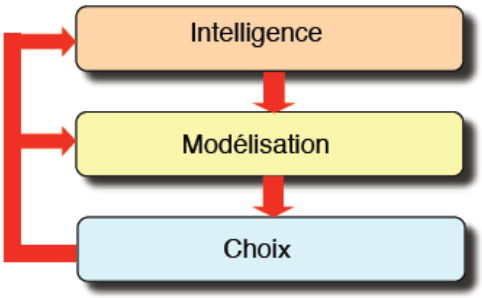

Figure 1.14 : Modèle de prise de décision IDC

c. Choix de décision créative

Définition 1.8 :

La créativité : capacité à développer des idées inédites, originales et utiles face aux problèmes à résoudre ou aux opportunités à saisir.

La créativité est basée sur trois éléments :

- expertise : capacités, connaissances, compétences ou expertises nécessaires à la créativité.
- ingéniosité : tous les aspects de la personnalité associés à la créativité, la capacité à employer des analogies ainsi que le don de voir le connu sous un nouveau jour.
- motivation : le souhait de travailler sur une tâche parce qu'elle suscite un intérêt, suppose une implication, apporte une satisfaction ou constitue un défi personnel

I.2.4.4 Gestion de risque

Toute décision est liée à des risques. Dans le processus décisionnel, le système essaie de définir la liste des risques et leurs probabilités d'apparition :

- si le risque est identifié, il devrait être accompagné d'un processus de surveillance
- si le risque est non identifié, il peut devenir un incident

Le processus décisionnel devrait inclure les étapes dans le tableau 1.2 pour annuler les effets de risques après la prise de décision.

Tableau 1.2 : Liste des étapes de traitement de risque

Etape	**Description**
L'identification des risques	Cette phase consiste à identifier et à structurer les risques opérationnels qui seront présentés dans un support de cartographie des risques
L'évaluation des risques	A partir de la cartographie des risques opérationnels établis, il convient d'effectuer une évaluation des risques identifiés - gravité : c'est l'impact maximum de l'exposition réelle ou potentielle aux situations de risque. - détection/Gestion : il s'agit de la capacité de l'entreprise à identifier et à réagir face aux évènements de risques. - occurrence : c'est la probabilité d'apparition des situations de risque.
La surveillance des risques	Cette phase correspond à la mise en place d'un dispositif de suivi et de contrôle du profil de risque de l'entreprise.

I.3 Dimension du système d'information

Définition 1.9 :

Le SI est un système d'interprétation d'un ensemble d'acteurs sociaux qui mémorisent et transforment des représentations via des technologies et des modes opératoires.

Selon cette définition, le système d'information se caractérise par trois dimensions :

- organisationnelle (acteurs)
- informationnelle
- technologique

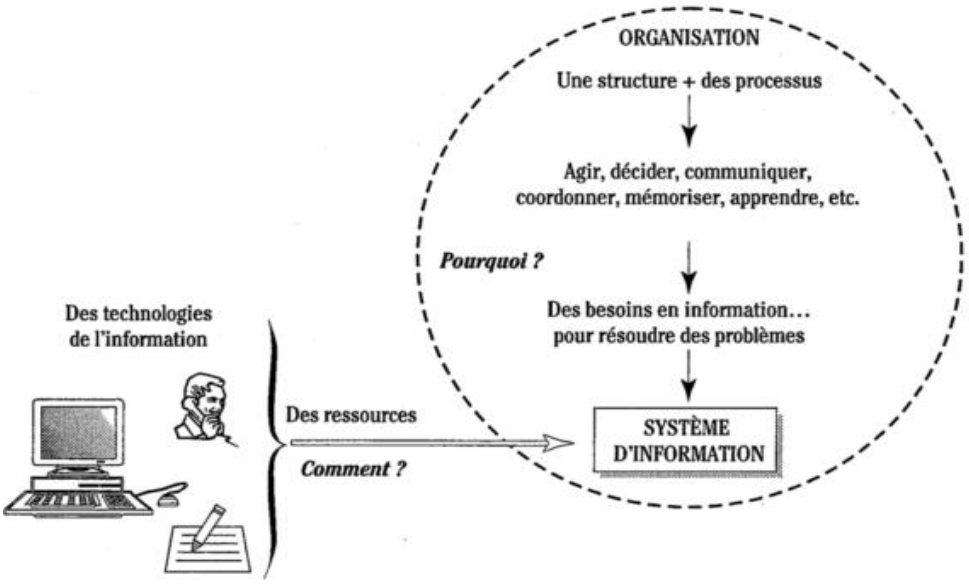

Figure 1.15 : Description de besoin du SI

Selon la figure 1.15, les systèmes d'information sont conçus pour aider l'organisation à prendre des décisions.
Pour représenter ces besoins de l'organisation, on a besoin d'identifier la dimension informationnelle de l'organisation selon les catégories suivantes :

- opérationnel
- ressources
- organisation

La dimension du SI est définie à l'aide des graphes qui relient chaque dimension listé dans chaque catégorie d'informations. Cette information permet de structurer et d'analyser l'effet de propagation d'un changement au sein d'une catégorie vers l'ensemble du système.

I.4 Outil de gestion du SI

L'objectif principal d'un outil de gestion de système d'information consiste à restituer l'information à la personne concernée sous une forme appropriée et au moment opportun. Il est estimé que près de 60% des systèmes mis en place dans le secteur privé ont échoué. Par « échoué », on entend :

- le système n'est pas utilisé par les utilisateurs supposés ne se le sont pas approprié
- le système n'est pas pertinent pour répondre aux préoccupations des utilisateurs qui ont donc recours à d'autres ressources
- le système ne contient pas l'information recherchée
- le projet de système, trop ambitieux, n'est pas arrivé à terme

De ce fait, les besoins principaux que l'on doit prendre en compte sont :

- simplicité de l'intégration
- évolutif (le traitement du métier ne dépend que des décisions des utilisateurs)
- Paramétrable en entrée et en sortie

I.4.1 Modèle d'un système d'information

En tenant compte du schéma fonctionnel d'un SI (Figure 1.1), le processus de gestion des incidents et des risques et le principe de génération de décision, on peut détailler le modèle de traitement d'un système d'information comme suit :

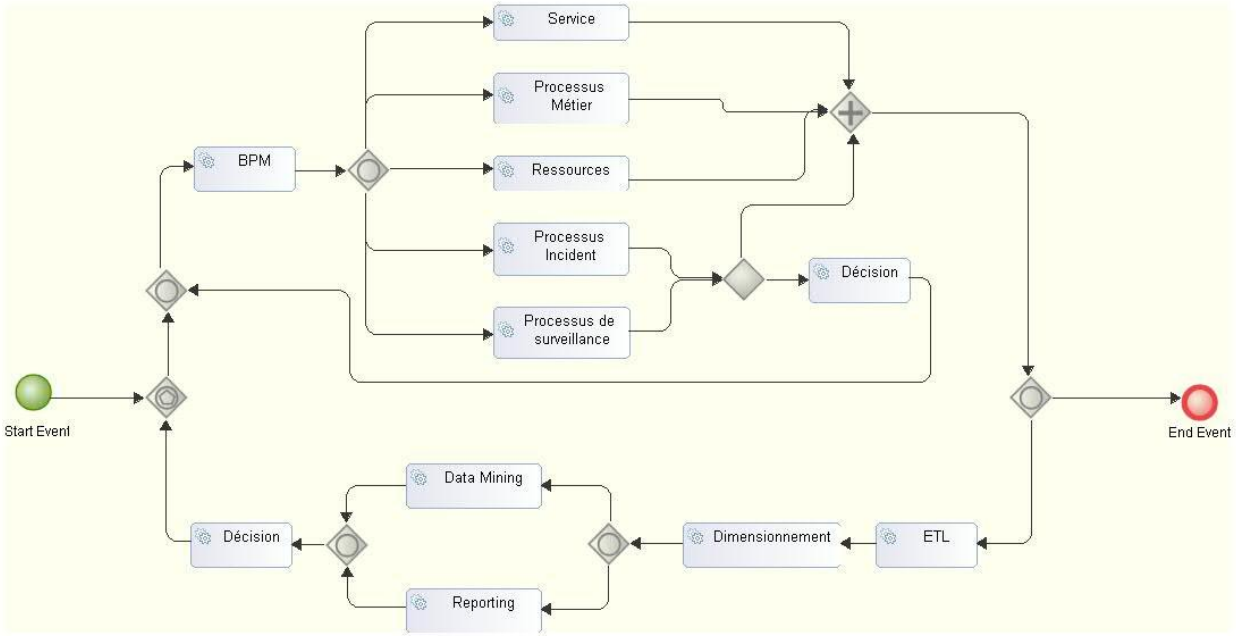

Figure 1.16 : Modèle d'un SI

Le type de décision mis en cause dans ce modèle concerne uniquement les SMO car il impacte directement la structure des traitements au niveau opérationnel.

Avec cette structure, le système est techniquement constitué :

- de gestionnaire de processus
- des applications basées sur des processus (métiers, incident, surveillance de risque)
- des applications clients / serveurs (réutilisable et modulaire)
- des applications de gestion des ressources (information dont la nature est statique)
- des matériels de communication et d'informatiques

La réalisation d'un tel modèle passe par plusieurs niveaux de conception. Certains blocs (Processus, Services, Report, ETL, Data mining) sont flexibles et d'autres sont rigides (BPM, Décision, Dimensionnement)
La conception commence par l'étude de l'existant et l'aspect technique de l'organisation. Cette étude permet de modéliser les traitements et les processus initiaux.

I.4.2 Contrainte de l'existant

La modélisation des données est un facteur clé de succès ou d'échec dans l'évolution des organisations. La phase d'étude de l'existant et des besoins est une phase essentielle à l'élaboration et la mise en place du système d'information. Elle doit aboutir à des spécifications générales qui décrivent les données manipulées, et les traitements à effectuer sur ces données.

Cette analyse permet de modéliser les besoins opérationnels du système :

- métier et activités de l'organisation
- ressources (humaine, matériels, documents, etc.)
- organigramme
- procédés
- produits
- clients
- fournisseurs
- processus

I.4.3 Solution orientée métier

Avec des différentes méthodes d'analyse standard du génie logiciel, le centre de toute la préoccupation est le métier de l'organisation et ses objectifs.
Cette démarche est appliquée à l'état de l'organisation à un moment donné par la séquence de traitements défini dans la figure 1.17 :

- définition des objectifs à atteindre
- décomposition en sous système
- identification des acteurs
- schématisation des interactions internes et externes
- modélisation des traitements
- modélisation des données
- planification et mise place

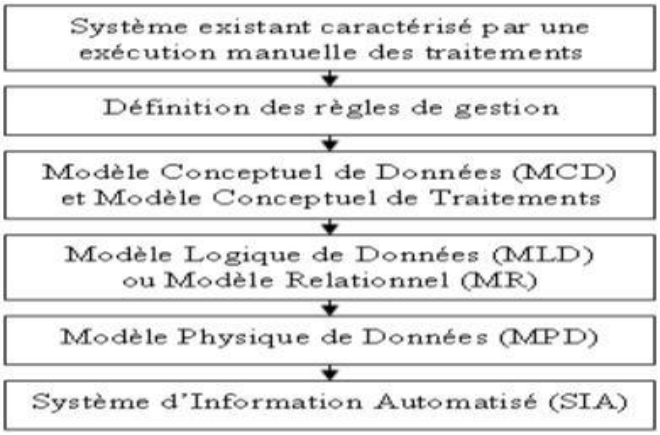

Figure 1.17 : Démarche de conception d'un SI avec la méthode d'analyse MERISE

I.5 Conclusion

L'ingénierie des SI reste un domaine complexe qui partant de l'analyse des besoins d'une application, aboutit à une solution logicielle fiable dans un système technologique cible choisi. Actuellement, la pratique industrielle de l'ingénierie des SI dans les entreprises industrielles rencontre diverses difficultés techniques, méthodologiques, organisationnelles et humaines.
Des difficultés qui constituent actuellement des goulots dans le processus de développement, engendrant des coûts considérables et menant dans beaucoup de cas à l'échec des projets SI. Il s'agit notamment :

- de la lenteur du développement des SI qui engendre des coûts considérables et mène parfois à la livraison de systèmes déjà obsolètes,
- de la nature statique de modélisation des données et des traitements,
- de l'insuffisance de l'analyse de l'impact des ressources technologiques dans la démarche,
- de l'insuffisance de la mise en valeur de la boucle de retour dans la conception. La solution est basée essentiellement au bloc opérationnel

Le SI est un moteur qui traite des informations pour produire d'autres informations. Cette boucle de traitement est dynamique selon le type de commande engendré par les décisions de l'organe de pilotage.

C'est sur ces hypothèses que se base donc la définition de notre problématique. Ainsi le travail de cette thèse a pour objectif de mettre en place un cadre méthodologique pour l'ingénierie des SI en abordant trois aspects :

- la spécification de différents blocs qui couvrent la complexité du SI (représentation des processus, des services de traitement des métiers, etc.) et favorisent l'échange d'information entre acteurs;
- la capitalisation et la réutilisation de savoir et de savoir-faire expérimentés et mis en œuvre dans des projets antérieurs. Cette démarche consiste à modéliser les patterns et les expériences du SI ;
- la modélisation d'un système ouvert permet de supporter l'évolution des processus au sein de l'organisation tout en gardant les gains des patterns au niveau composant et au niveau SI

CHAPITRE 2 : Design Pattern

II.1 Introduction

L'idée sur le design pattern a été introduite par Alexander dans le domaine de l'architecture et formalisée dans le domaine de génie logiciel par la publication du livre « *Design Patterns - Elements of Reusable Object-Oriented Software* » en 1995. Le fait que les patrons capitalisent un savoir-faire consensuel sur un domaine donné, ils se sont développés au sein de la communauté du génie logiciel et en particulier parmi les personnes s'intéressant aux approches orientées objet et à la réutilisation.

Dans le domaine de l'ingénierie logicielle, il est primordial la conception de l'architecture du logiciel, c'est-à-dire sa décomposition en sous-systèmes, modules et composantes individuelles qui la forment. Représenter précisément les éléments structuraux et comportementaux de cette architecture à l'aide de notations normalisées.

Cette conception nécessite la maitrise de chaque élément (entité logiciel) qui le constitue et ses interactions. Les patterns sont des solutions et des outils fiables pour nous aider à choisir les composants du SI ainsi que la modalité qui gère sa structure.

II.2 Design Pattern et l'expérience

Définition 2.1:

"*Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice*".

Chaque patron décrit un problème qui se produit encore et encore dans notre environnement, et ensuite décrit le cœur d'une solution à ce problème de manière à ce qu'on puisse utiliser cette solution plus d'un million de fois, sans jamais le faire deux fois exactement de la même manière.

Définition 2.2 :

La réutilisation se définit comme une approche de développement de systèmes, qui propose notamment de construire un système nouveau à partir de composants logiciels sur étagère (*COTS, Commercial Off The Shelf*). Elle s'oppose ainsi aux approches traditionnelles de développement, dans lesquelles la construction d'un nouveau système part de rien (*from scratch*) et nécessite de réinventer de grandes parties d'applications à chaque fois.

Les design patterns permettent d'apporter une solution éprouvée, efficace et réutilisable, il est le résultat d'une meilleure compréhension du design complet du problème. Ils sont donc le fruit d'une expérience : un patron décrit une situation fréquente et une réponse éprouvée à cette situation.

II.2.1 Savoir faire

La notion sous-jacente aux patrons de conception est aussi connexe à la notion de savoir-faire. Elle englobe en particulier l'idée d'un plus haut niveau de compréhension élaboré à partir d'interactions entre composants primaires.

Pour illustrer cette définition, on peut comparer l'apprentissage de la conception de logiciels et l'apprentissage du jeu d'échecs, où trois niveaux de maîtrise sont bien identifiés :

Tableau 2.1 : Niveau d'apprentissage de jeux d'échecs

Niveau	**Jeux d'échecs**	**Programmation**
1 : Apprendre les règles de base :	Apprendre le nom des pièces, les mouvements légaux, l'orientation et la géométrie de l'échiquier, etc.	Apprendre des algorithmes de base, des structures de données et des langages de programmation
2 : Apprendre les principes	Apprendre la valeur relative de certaines pièces (comme la reine), la valeur stratégique des cases centrales de l'échiquier	Apprendre la connaissance des principes d'organisation du logiciel (modularité, orientation objet, généricité, etc.).
3 : Apprendre les motifs récurrents	Pour devenir un maître d'échecs, il faut en effet étudier le jeu des autres maîtres : les parties d'échecs disputées par des grands maîtres contiennent en effet des patrons qui devraient être compris, mémorisés, et appliqués de manière répétitive	Les patterns s'acquièrent en étudiant les applications déjà existantes

II.2.2 Pattern prouvé

Un pattern est un ensemble de variables, d'événements et d'invariants qui vont permettre de modéliser une notion que l'on veut voir figurer dans le système que l'on construit (système cible).

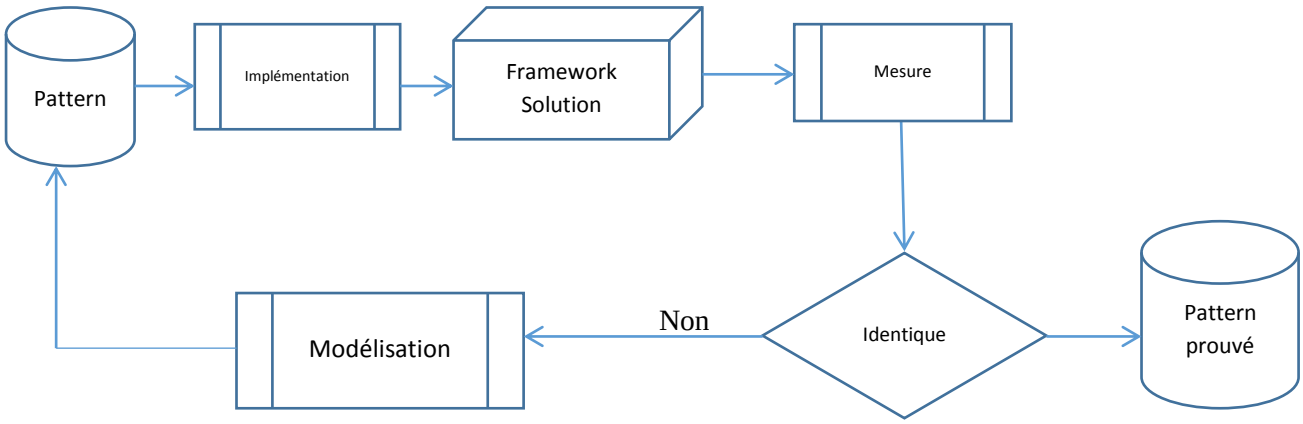

Figure 2.1 : Procédure d'approbation de pattern

En utilisant le processus défini par la figure 2.1, un pattern est prouvé lorsque les événements décrits dans ce pattern vérifient les propriétés invariantes exprimées sur le système cible quels que soient les nombres des instanciations.

II.2.3 Pattern et UML

La structure et la représentation font partie des propriétés d'un pattern. UML en tant que langage de modélisation nous offre le moyen standard de présenter les participants dans la solution ainsi que leurs interactions.

UML permet de présenter un modèle sous plusieurs diagrammes mais en général pour les patterns, les diagrammes de classe, de séquence et de cas d'utilisation sont les plus utilisés.

La description de la solution finale en termes de classe et d'interaction (diagramme de classe et séquence) est la première préoccupation des utilisateurs de pattern pendant son implémentation. Tandis que le cas d'utilisation est utile pour choisir lequel est intéressant pendant la capture de besoin.

II.3 Pattern de Gang of Four

Les 23 patterns définis par Gamma et ses équipes restent une référence pour toute forme de pattern. La plupart des Framework implémentent les solutions définies dans ces patterns.
Ils sont groupés en 3 catégories :

- création :
    - description de la manière dont un objet ou un ensemble d'objets peuvent être créés, initialisés, et configurés
    - isolation du code relatif à la création, à l'initialisation afin de rendre l'application indépendante de ces aspects
- structure
    - description de la manière dont doivent être connectés des objets de l'application afin de rendre ces connections indépendantes des évolutions futures de l'application
    - découplage de l'interface et de l'implémentation de classes et d'objets
- comportement.
    - description de comportements d'interaction entre objets
    - gestion des interactions dynamiques entre des classes et des objets

Tableau 2.2 : Liste des Pattern de Gang of Four

Catégorie	**Pattern**	**Motivation**
Création	Abstract factory	Fournir une interface pour la création de famille d'objets sans spécifier les classes concrètes
	Builder	Séparer la construction d'un objet complexe de sa représentation. Le même procédé de construction peut donc créer différentes représentations
	Factory Method	Définir une interface pour la création d'objet, en laissant les sous classes décider quelle classe instancier. (Laisser le travail d'instanciation aux sous classes)
	Prototype	Spécifier le type d'objet à créer en utilisant une instance d'un prototype, et créer de nouveaux objets en copiant ce prototype

	Singleton	S'assurer qu'une classe n'ait qu'une instance et lui fournir un point d'accès global
Structure	Adapter	Convertir l'interface d'une classe dans une autre interface compatible avec d'autres clients
	Bridge	Séparer une abstraction de son implémentation dans le but que chacune puisse être modifiée indépendamment
	Composite	Composer des objets en structure d'arbre hiérarchisé
	Decorator	Ajouter dynamiquement des responsabilités à un objet
	Facade	Proposer une interface commune à un ensemble d'interfaces d'un sous-système (améliorer l'utilisation d'un sous-système).
	Flyweight	Utiliser le partage afin de supporter un large panel d'objets bas niveau
	Proxy	Proposer un clone ou un point d'entrée à un objet pour contrôler son accès
Comportement	Chain of responsibility	Permettre d'associer l'émetteur et le récepteur d'une requête en donnant à plus d'un objet la chance de manipuler la requête (enchaîner des objets et parcourir cette chaîne avec la requête jusqu'à trouver quelqu'un capable de la manipuler)
	Command	Encapsuler une requête dans un objet
	Interpreter	Proposer un langage, définir une représentation pour sa grammaire et fournir un interprète capable de manipuler cette grammaire
	Iterator	Proposer séquentiellement une façon d'accéder aux éléments d'un objet
	Mediator	Définir un objet qui encapsule la façon dont interagit un autre ensemble d'objets
	Memento	Sans toucher à l'encapsulation, capturer et extraire l'état d'un objet de manière à ce que cet objet puisse retrouver son état initial
	Observer	Proposer une dépendance entre objets de manière à mettre à jour automatiquement toutes ces dépendances lors du changement d'état d'un objet
	State	Permettre à un objet de modifier son comportement lorsque son état interne change.
	Strategy	Définir une famille d'algorithmes, l'encapsuler, la rendre interchangeable, et permettre à chaque algorithme de varier indépendamment du client qui l'utilise
	Template method	Définir le squelette d'un algorithme en une opération et allouer certaines parties aux sous classes (les sous classes redéfinissent certaines parties de l'algorithme sans changer la structure principale)
	Visitor	Permettre de définir une nouvelle opération sans changement de classe des éléments utilisés