

Faculté des Sciences de Tunis
Département des sciences de l'informatique

**CONCEPTION
DES
SYSTÈMES
D'INFORMATIONS**

A. Abdellatif

*Année universitaire
2007/2008*

OBJECTIFS DU COURS

- ◆ Comprendre la notion de système d'information et sa position dans l'entreprise
- ◆ Découvrir les différents types de systèmes d'information et leurs domaines d'utilisation
- ◆ Découvrir les principes et les caractéristiques des méthodes de conception de SI et leur évolution
- ◆ Découvrir les cycles de développement de SI et les différents modèles existants
- ◆ Étudier une méthode de conception de 2ème génération : Merise
- ◆ Étudier l'évolution d'une méthode de 2ème génération : Merise 2
- ◆ Étudier quelques méthodes de conception de 3ème génération (objet)
- ◆ Découvrir les perspectives en matière de conception de SI

SOMMAIRE GÉNÉRAL

1. INTRODUCTION AUX SYSTÈMES D'INFORMATION.....	6
1.1 NOTION DE SYSTÈME.....	6
1.2 DÉCOMPOSITION D'UN SYSTÈME EN SOUS-SYSTÈMES.....	9
1.3 NOTION DE SYSTÈME D'INFORMATION.....	11
1.4 TYPES DE SYSTÈMES D'INFORMATION.....	14
1.4.1 <i>SI opérant</i>	15
1.4.2 <i>SI de pilotage</i>	15
1.5 DOMAINES D'UTILISATION DES DIFFÉRENTS TYPES DE SI.....	16
1.6 FONCTIONS DU SYSTÈME D'INFORMATION.....	17
2. MÉTHODES DE CONCEPTION DES SYSTÈMES D'INFORMATION.....	18
2.1 INTRODUCTION.....	18
2.2 OBJECTIFS DES MÉTHODES DE CONCEPTION.....	19
2.3 CYCLES DE CONCEPTION ET DE DÉVELOPPEMENT DE SI.....	20
2.3.1 <i>Introduction</i>	20
2.3.2 <i>Modèle en cascade</i>	21
2.3.3 <i>Modèle en V</i>	24
2.3.4 <i>Modèle en spirale</i>	27
2.3.5 <i>Modèle tridimensionnel</i>	30
2.4 ARCHITECTURE ANSI/SPARC.....	36
2.4.1 <i>NIVEAU CONCEPTUEL</i>	37
2.4.2 <i>NIVEAU INTERNE</i>	38
2.4.3 <i>NIVEAU EXTERNE</i>	39
2.5 HISTORIQUE DES MÉTHODES DE CONCEPTION DE SI.....	40
2.5.1 <i>Critères de classification des méthodes de conception de SI</i>	40
2.5.2 <i>Classification des méthodes de conception de SI</i>	42
2.6 MÉTHODES ANALYTIQUES OU CARTÉSIENNES.....	43
2.6.1 <i>Principes</i>	43
2.6.2 <i>Exemples</i>	44
2.6.3 <i>Avantages et inconvénients</i>	44
2.7 MÉTHODES SYSTÉMIQUES.....	46
2.7.1 <i>Principes</i>	46
2.7.2 <i>Exemples</i>	47
2.7.3 <i>Avantages et inconvénients</i>	48
2.8 MÉTHODES ORIENTÉES OBJET.....	49
2.8.1 <i>Principes</i>	49
2.8.2 <i>Exemples</i>	50
2.8.3 <i>Avantages et inconvénients</i>	50
A RETENIR.....	52
3. ÉTUDE D'UNE MÉTHODE SYSTÉMIQUE : MERISE.....	54
3.1 INTRODUCTION.....	54
3.2 LES ÉTAPES.....	55
3.2.1 <i>Schéma directeur</i>	56
3.2.2 <i>Étude préalable</i>	57
3.2.3 <i>Étude détaillée</i>	62
3.2.4 <i>Réalisation</i>	64
3.3 LA MODÉLISATION.....	66
3.4 MODÉLISATION DES DONNÉES.....	70
3.4.1 <i>Modèle conceptuel de données</i>	70

3.4.2	Modèle logique de données (MLD).....	86
3.4.3	Modèle physique de données.....	91
3.5	MODÉLISATION DES TRAITEMENTS	93
3.5.1	Modèle conceptuel de traitements.....	93
3.5.2	Modèle organisationnel de traitements.....	107
3.6	RÉSUMÉ	113
4.	MERISE 2 : EXTENSION DE MERISE.....	114
4.1	INTRODUCTION	114
4.2	NOTION DE LIEN IDENTIFIANT	115
4.2.1	Principe au niveau conceptuel.....	115
4.2.2	Exemples	115
4.2.3	Traduction d'un lien identifiant au niveau MLD.....	116
4.3	NOTION D'ASSOCIATION D'ASSOCIATIONS	117
4.3.1	Principe au niveau conceptuel.....	117
4.3.2	Exemple.....	118
4.3.3	Traduction d'une association d'associations au niveau MLD.....	118
4.4	NOTION D'HÉRITAGE.....	120
4.4.1	Principe au niveau conceptuel.....	120
4.4.2	Exemple.....	121
4.4.3	Traduction de l'héritage au niveau MLD	122
5.	MÉTHODES DE CONCEPTION ORIENTÉES OBJET.....	124
5.1	INTRODUCTION	124
5.2	GÉNÉRALITÉS	126
5.2.1	Concepts de base.....	126
5.2.2	Les trois dimensions du SI	130
5.3	MÉTHODE OMT.....	132
5.3.1	Présentation générale	132
5.3.2	Principes de la méthode.....	133
5.3.3	Démarche méthodologique	134
5.3.4	Modélisation.....	144
5.4	AUTRES MÉTHODES OBJET	152
5.4.1	Méthode OOD.....	152
5.4.2	Méthode HOOD	158
5.4.3	Méthode OOA	160
5.4.4	Méthode OOM.....	163
6.	CONCLUSION ET PERSPECTIVES	167

BIBLIOGRAPHIE

◆ **Systèmes d'information : structuration, modélisation et communication**

J-C Courbon

Inter Édition 1993

◆ **L'essentiel sur Merise**

Dominique Dionisi

Ed. Eyrolles 1993

◆ **Objets : du C++ à Merise Objet**

M. Bouzeghoub, G. Gardarin, P. Valduriez

Ed. Eyrolles 1994

◆ **OMT : Modélisation et conception orientées objet**

James Rumbaugh et al.

Ed. Prentice Hall et Masson 1995

◆ **Analyse et conception orientées objet**

G. Booch

Addison-Wasley 1994

◆ **Modélisation objet avec UML**

P.-A. Muller

Eyrolles 2005

◆ **RAD : une méthode pour développer plus vite**

J. Hugues, B. Leblanc, C. Morley

InterEditions 1996

1. INTRODUCTION AUX SYSTÈMES D'INFORMATION

1.1 Notion de système

Définition :

Un système est un **tout** constitué **d'éléments** unis par des **relations**, ces éléments et ces relations étant munis de **propriétés**.

Description d'un système :

- ◆ Déterminer les éléments et les relations
- ◆ Déterminer les propriétés des éléments et des relations
- ◆ Déterminer les valeurs que peuvent prendre les propriétés
- ◆ Décrire l'activité du système
- ◆ Décrire son organisation

Exemple :

Système : entreprise

Éléments : employés, services, articles, entrepôt....

Propriétés des éléments : Nom et matricule d'employé, référence d'article

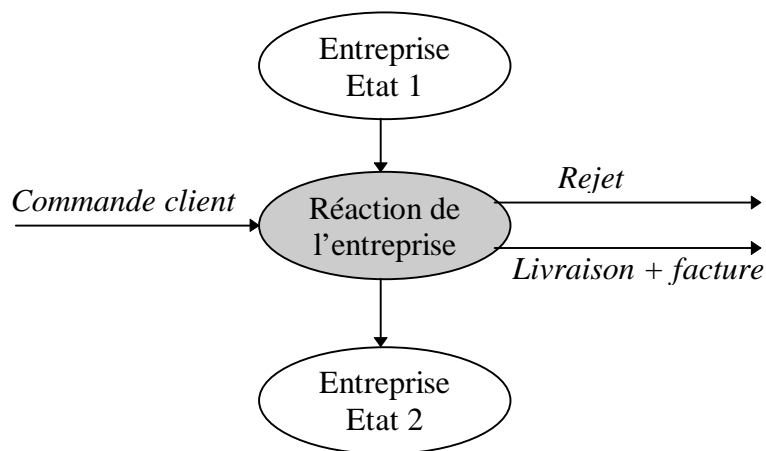
Relations : un employé **est rattaché** à un service, un article **est stocké dans** un entrepôt

Propriétés des relations : Date d'entrée dans le service, quantité stockée

État d'un système :

- ◆ L'état d'un système est déterminé par l'ensemble des valeurs prises par les propriétés des éléments et des relations.
- ◆ Changement de valeurs \implies changement d'état du système

Exemple :



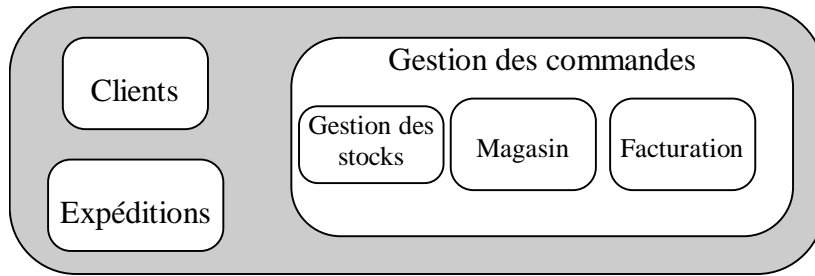
Environnement d'un système :

- ◆ Un système subit de la part de son environnement un ensemble de contraintes.
- ◆ Ces contraintes obligent le système à réagir en déclenchant des activités tendant à le ramener à un état stationnaire.

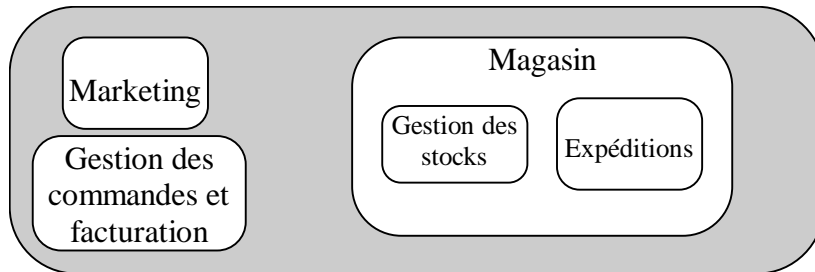
1.2 Décomposition d'un système en sous-systèmes

- ◆ Tout système complexe peut être décomposé en sous-systèmes.
- ◆ Les sous-systèmes d'un même système sont unis par des relations.
- ◆ Les sous-systèmes sont considérés comme des éléments du système.
- ◆ Pour un même système, il existe plusieurs décompositions possibles en sous-systèmes.
- ◆ La qualité de la représentation du système dépend de sa bonne décomposition.

Exemple :



Première décomposition



Deuxième décomposition

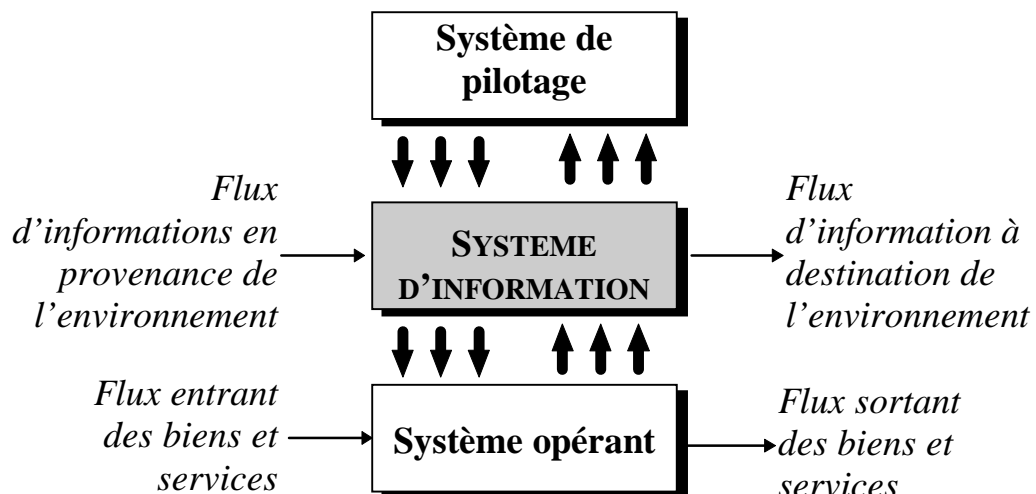
1.3 Notion de système d'information

Définitions :

◆ « Le système d'information est **une représentation** possible de n'importe quel système, notamment tout système humain organisé. »

◆ « Le système d'information est le **véhicule de la communication** dans l'entreprise. Cette communication possède un **langage** dont les mots sont les **données**. »

◆ « Le système d'information est le système de **couplage** entre le **système opérant** et le **système de pilotage**. »

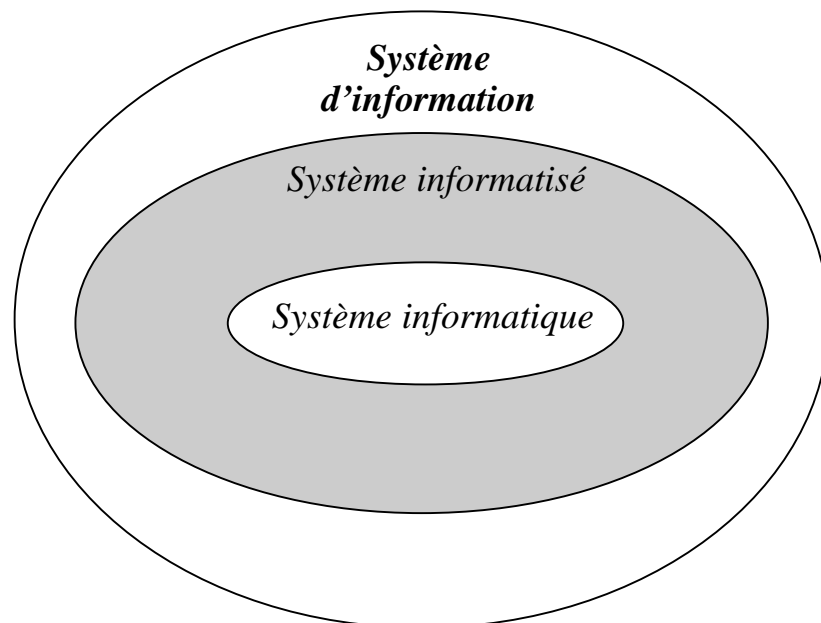


- ◆ Le système d'information sert comme moyen de communication entre
 - ◇ l'entreprise et l'environnement extérieur
 - ◇ le système opérant et le système de pilotage

Remarque :

A ne pas confondre

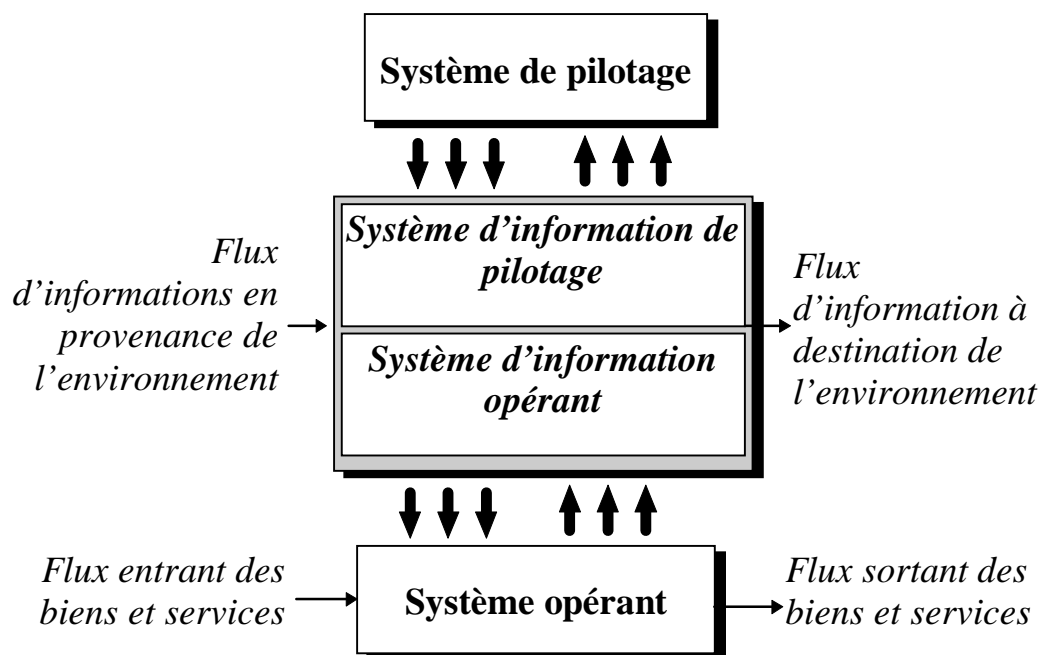
- ◇ **Systeme d'information,**
- ◇ **Systeme informatisé et**
- ◇ **Systeme informatique.**



1.4 Types de systèmes d'information

Le système d'information peut être décomposé en deux types :

- ◆ Système d'information **opérant**
- ◆ Système d'information de **pilotage**



1.4.1 SI opérant

Il prend en charge la gestion courante de l'entreprise :

- ◆ gestion du personnel
- ◆ gestion de la production
- ◆ gestion des stocks
- ◆ facturation
- ◆ comptabilité

1.4.2 SI de pilotage

Il prend en charge le pilotage et le traitement de gestion par exception :

- ◆ états statistiques
- ◆ historique
- ◆ décisions
- ◆ plans à long terme

Remarque :

La frontière entre système d'information opérant et système d'information de pilotage n'est pas toujours claire.

Il existe une zone commune correspondant aux besoins **tactiques**.

1.5 Domaines d'utilisation des différents types de SI

Trois niveaux de management utilisent les deux types de systèmes d'information :

- ◆ Niveau de management Stratégique
- ◆ Niveau de management Tactique
- ◆ Niveau de management Opérationnel

<i>Niveau de management</i>	<i>Système d'information</i>	
<i>Stratégique</i>	<i>De pilotage</i>	Stratégique
<i>Tactique</i>		Tactique
<i>Opérationnel</i>	<i>Opérant</i>	Tactique
		Opérationnel

1.6 Fonctions du système d'information

Un système d'information doit assurer les fonctions suivantes :

- ◆ **Saisie** : Saisie des données faisant partie du SI pour qu'elles aient une existence réelle.
- ◆ **Mémorisation** : Permet de retrouver la donnée ultérieurement (persistance)
- ◆ **Traitement** : Permet d'accéder aux données, les mettre à jour et les mettre en forme.
- ◆ **Communication** : Permet la communication entre le système d'information et son environnement ainsi qu'avec le système opérant et le système de pilotage.

2. MÉTHODES DE CONCEPTION DES SYSTÈMES D'INFORMATION

2.1 Introduction

- ◆ Systèmes d'information de plus en plus **complexes**
- ◆ **Nombre de participants** à la conception du SI de plus en plus important (quelques dizaines)
- ◆ **Durée de conception** et de mise en oeuvre d'un SI de plus en plus importante (quelques années)
- ◆ Importance des **enjeux financiers** et des **risques**



Nécessité d'une méthode de conception et de développement des systèmes d'information.

2.2 Objectifs des méthodes de conception

- ◆ Permettre la description des SI à l'aide de **modèles**, selon une **démarche** (étapes) et des moyens de **contrôle qualité**.
- ◆ Aider à **réaliser le système informatisé** correspondant au système d'information.
- ◆ Diminuer les **coûts** et les **risques** des projets d'informatisation.
- ◆ Rendre l'activité de conception et de développement de SI une **activité d'ingénierie** au même titre que le génie mécanique, le génie civil, ...
- ◆ Permettre à l'équipe de conception et de développement de disposer d'un **vocabulaire standard**.

2.3 Cycles de conception et de développement de SI

2.3.1 Introduction

- ◆ Un projet de conception et de développement d'un SI est composé d'**étapes**.

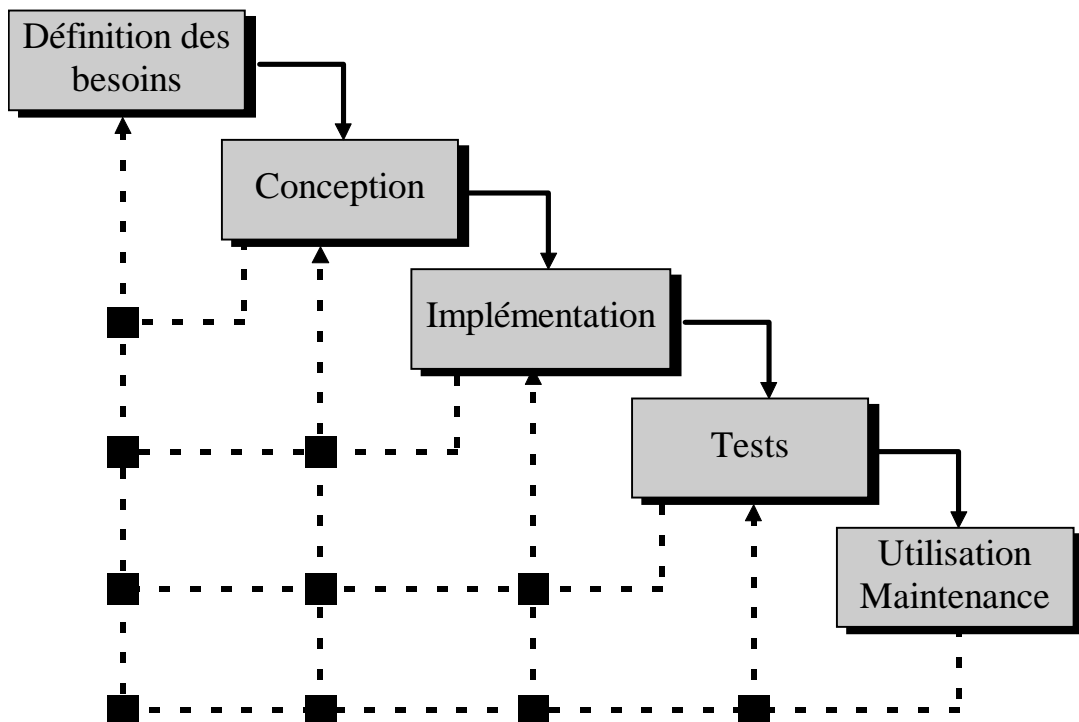
- ◆ Le découpage du projet en étapes et l'organisation de ces étapes varient selon le **modèle** utilisé.

- ◆ Il existe quatre modèles :
 - ◇ Modèle en **cascade**
 - ◇ Modèle en **V**
 - ◇ Modèle en **spirale**
 - ◇ Modèle **tridimensionnel**

2.3.2 Modèle en cascade

Principes :

- ◆ Modèle utilisé pour la conception et le développement de la première génération d'applications informatiques : années 60 et début des années 70.
- ◆ Cinq étapes :
 - ◇ Définition des besoins
 - ◇ Conception
 - ◇ Implémentation
 - ◇ Tests
 - ◇ Utilisation et maintenance



Les étapes :

- ◆ **Définition des besoins** : Automatisation d'une activité selon les principes d'identification des *entrées*, des *sorties* et des *transformations* à réaliser.
- ◆ **Conception** : Spécification technique détaillée (description de fichiers, d'algorithmes et d'états de sortie).
- ◆ **Implémentation** : Codage.
- ◆ **Tests** : Mise au point et validation.
- ◆ **Utilisation et maintenance** : Exploitation des applications et leur maintenance en cas de besoin.

Itérations sur les étapes :

- ◆ Les itérations sur les étapes sont théoriquement possibles d'une étape vers n'importe quelle étape précédente.
- ◆ Ce retour n'a pas toujours de sens (de l'étape de tests ou d'utilisation vers la définition des besoins par exemple).

Inconvénients du modèle :

- ◆ Incapacité à prendre en charge des systèmes complexes comportant un grand nombre d'applications interagissant les unes avec les autres.
- ◆ Absence de phase de conception générale : passage direct de l'analyse des besoins à une phase de conception détaillée (technique).
- ◆ La phase de test s'applique à la totalité de l'application et n'englobe pas la validation par rapport aux besoins).
- ◆ Il n'existe pas de modélisation du système d'information.

2.3.3 Modèle en V

Principes :

- ◆ C'est une variante du modèle en cascade.
- ◆ Supporte la notion de système et de sous-système.
- ◆ Le système est décomposé en sous-systèmes permettant la conception des systèmes complexes.
- ◆ Sept étapes :
 - ◇ Définition des besoins
 - ◇ Conception du système
 - ◇ Conception des composants
 - ◇ Codage des composants
 - ◇ Test des composants
 - ◇ Test du système
 - ◇ Validation.

Inconvénients du modèle :

- ◆ La validation par rapport aux besoins intervient assez tard dans le cycle.

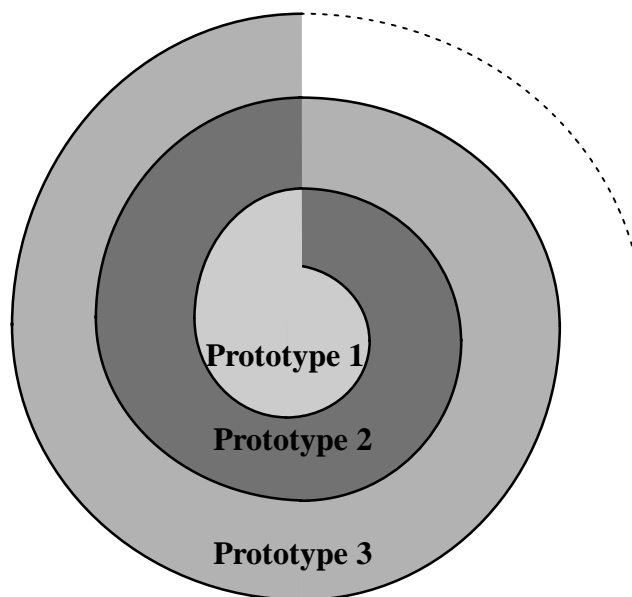
Avantages du modèle :

- ◆ Le découpage du système en sous-système permet d'avoir une conception et un développement modulaire.
- ◆ Convient aux systèmes complexes.

2.3.4 Modèle en spirale

Principes :

- ◆ C'est le modèle le plus récent.
- ◆ Répond aux lacunes de validation du modèle en V.
- ◆ Le développement du SI se fait par une série de **prototypes** correspondant à des sous-systèmes représentatifs.
- ◆ La **validation** se fait au **plus tôt possible**, par sous-système, par rapport aux besoins fonctionnels, , aux contraintes matérielles ou logicielles et aux considérations économiques ou stratégiques.
- ◆ Permet de montrer la **validité de la compréhension** que le concepteur a de la réalité et des besoins des utilisateurs et le **bien fondé** des choix techniques.
- ◆ Adapté au développement RAD (Rapid Application Development).



Les étapes :

Pour chaque sous-ensemble :

Jusqu'à satisfaction des utilisateurs :

- ◆ **Définition des besoins** : Recensement des besoins pour la conception du sous-système.
- ◆ **Conception** : Spécification détaillée du sous-système.
- ◆ **Implémentation** : Réalisation par étapes du sous-système (interface utilisateur, Contrôles, accès base de données, ...)
- ◆ **Test** : Test des composantes du sous-système.
- ◆ **Validation** : Vérification que le sous-système construit correspond bien aux besoins des utilisateurs.

Inconvénients du modèle :

- ◆ Ne convient que pour les projets qui peuvent être découpés en sous projets (sous-systèmes).
- ◆ Le coût pourrait être élevé.
- ◆ Ne convient que pour les applications dans laquelle l'interface utilisateur est prépondérante.

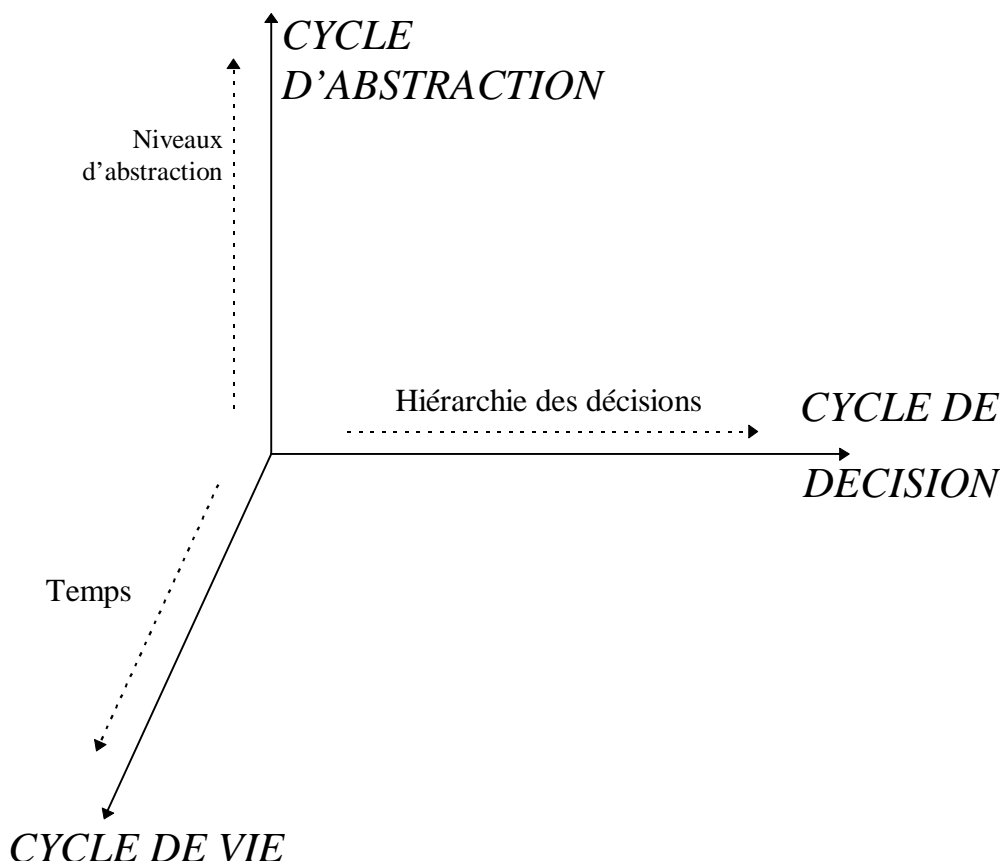
Avantages du modèle :

- ◆ Validation concrète et sûre par les utilisateurs.
- ◆ Validation au plus tôt.
- ◆ Réduction du temps d'attente des utilisateurs (recensement des besoins et livraison des applications) et maintien de leur motivation.
- ◆ Forte implication des utilisateurs.

2.3.5 Modèle tridimensionnel

Principes :

- ◆ C'est le modèle introduit par la méthode Merise.
- ◆ Le développement du système d'information se fait suivant trois axes appelés **cycles** :
 - ◇ **Cycle de vie** : décrit les différentes étapes correspondant au cycle de vie du système d'information.
 - ◇ **Cycle de décision** : décrit le cycle de développement correspondant au cycle de vie du projet.
 - ◇ **Cycle d'abstraction** : comprend les niveaux de description du système d'information.



Le cycle de vie :

- ◆ Correspond à la vie du système d'information, depuis sa conception jusqu'à l'exploitation en passant par sa naissance, sa maturité et sa maintenance.
- ◆ Comprend trois périodes :
 - ◇ Conception : aboutit à la description détaillée des spécifications fonctionnelles et techniques du système.
 - ◇ Réalisation : production des programmes et structures de données correspondant aux spécifications détaillées.
 - ◇ Maintenance : adaptation du système à l'évolution de son environnement.
- ◆ Un bouleversement profond de l'organisation et de son environnement conduit à recommencer un cycle de vie : conception, réalisation, maintenance.

Exemples :

- Montée en charge du volume de données ou des transactions.
- Changement technologique (matériel ou logiciel)
- Restructuration organique : passage d'une organisation centralisée à une organisation décentralisée.

Le cycle de décision :

- ◆ Correspond aux choix qui doivent être faits durant le cycle de vie du système d'information.
- ◆ A travers ce cycle, l'organisation s'assure que le système correspond aux objectifs.
- ◆ Types de décisions :
 - ◇ Décisions de gestion : Objectifs, orientations, règles de gestions, ...
 - ◇ Décisions organisationnelles : Choix d'organisation, répartition des tâches
 - ◇ Décisions techniques : Choix techniques (SE, SGBD, outils de développement, Bureautique, architecture, ...).
 - ◇ Orientation de la gestion du projet : Ressources allouées, priorités de développement, planning d'avancement, ...

Le cycle d'abstraction :

- ◆ Correspond aux différents niveaux permettant la description et la spécification du système d'information.
- ◆ Trois niveaux d'abstraction :
 - ◇ Niveau conceptuel : C'est le niveau d'abstraction le plus élevé. Il comprend les éléments les plus stables. Il décrit les classes d'objets et les règles significatives en fonction des objectifs fixés par les décideurs.
 - ◇ Niveau logique / organisationnel : Représente les ressources utilisées pour supporter les descriptions du niveau conceptuel.
 - ◇ Niveau physique / opérationnel : Donne une représentation physique des données et opérationnelle des traitements en tenant compte des contraintes et choix techniques.

Niveaux de description	Données	Traitements
Conceptuel (déterminé par des choix de gestion)	Représentation conceptuelle des données sous forme d'entités et d'associations inter-entités	Représentation conceptuelle des traitements sous forme d'opérations déclenchées par des événements
Logique / organisationnel (déterminé par des choix d'organisation)	Description logique des données en fonction des schémas disponibles (fichiers classiques, schémas hiérarchique, réseau ou relationnel).	Simulation organisationnelle des traitements intégrant les ressources hommes, machines et interactions
Physique / opérationnel (déterminé par des choix techniques)	Représentation physique des données en fonction de la fréquence d'accès, des temps de réponses attendus, des choix de répartition, ...	Représentation opérationnelle des traitements : applicatifs, et communication en fonction des architectures, configurations, langages et environnements techniques retenus.

Inconvénients du modèle :

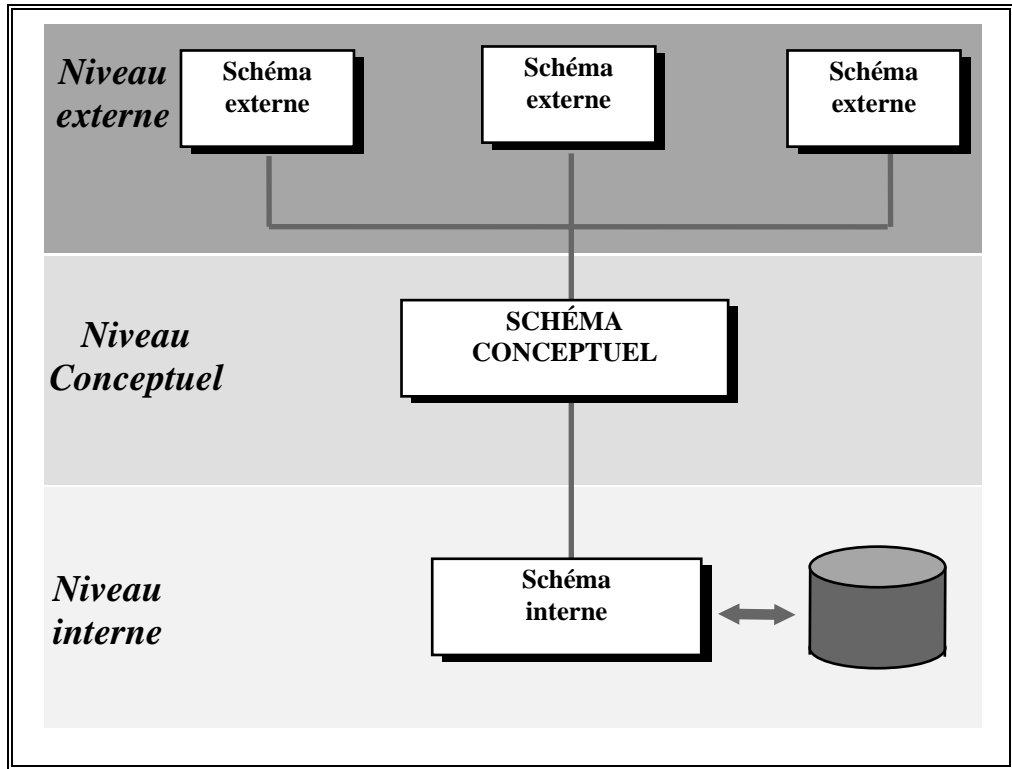
- ◆ Manque de formalisation du cycle de vie (critères qui caractérisent ce cycle de vie).
- ◆ Manque de sémantique pour chaque plan formé par une paire d'axes.

Avantages du modèle :

- ◆ Le point fort de ce modèle réside dans le cycle d'abstraction. Il permet d'avoir une indépendance entre la solution conceptuelle et la solution technique.
- ◆ CE modèle permet une meilleure portabilité et une plus grande évolutivité du système d'information.

2.4 Architecture ANSI/SPARC

- ◆ En 1969 un groupe de normalisation a été créé pour étudier l'impact des SGBD sur les systèmes d'information.
- ◆ En 1975 publication du rapport ANSI/X3/SPARC : proposition de **trois niveaux de description de données** :
 - ◇ Niveau conceptuel,
 - ◇ Niveau interne,
 - ◇ Niveau externe.



2.4.1 NIVEAU CONCEPTUEL

Correspond à la structure **sémantique** des données sans soucis d'implémentation.



SCHÉMA CONCEPTUEL :

Définit :

- ◇ Les types de données élémentaires qui définissent les entités
- ◇ Les entités
- ◇ Les associations entre les entités
- ◇ Les règles que suivront les données au cours de leur vie (contraintes d'intégrité).

2.4.2 NIVEAU INTERNE

Correspond à la structure de stockage supportant les données. Il permet de décrire les données telles qu'elles sont stockées en machine.



SCHEMA PHYSIQUE :

Définit :

- ◇ Les fichiers contenant les données (Nom, localisation, organisation, etc.)
- ◇ Les enregistrements de ces fichiers (longueur, champs composants, etc.)
- ◇ Les chemins d'accès à ces fichiers (index, chaînage, fichiers inverses, etc.)

2.4.3 NIVEAU EXTERNE

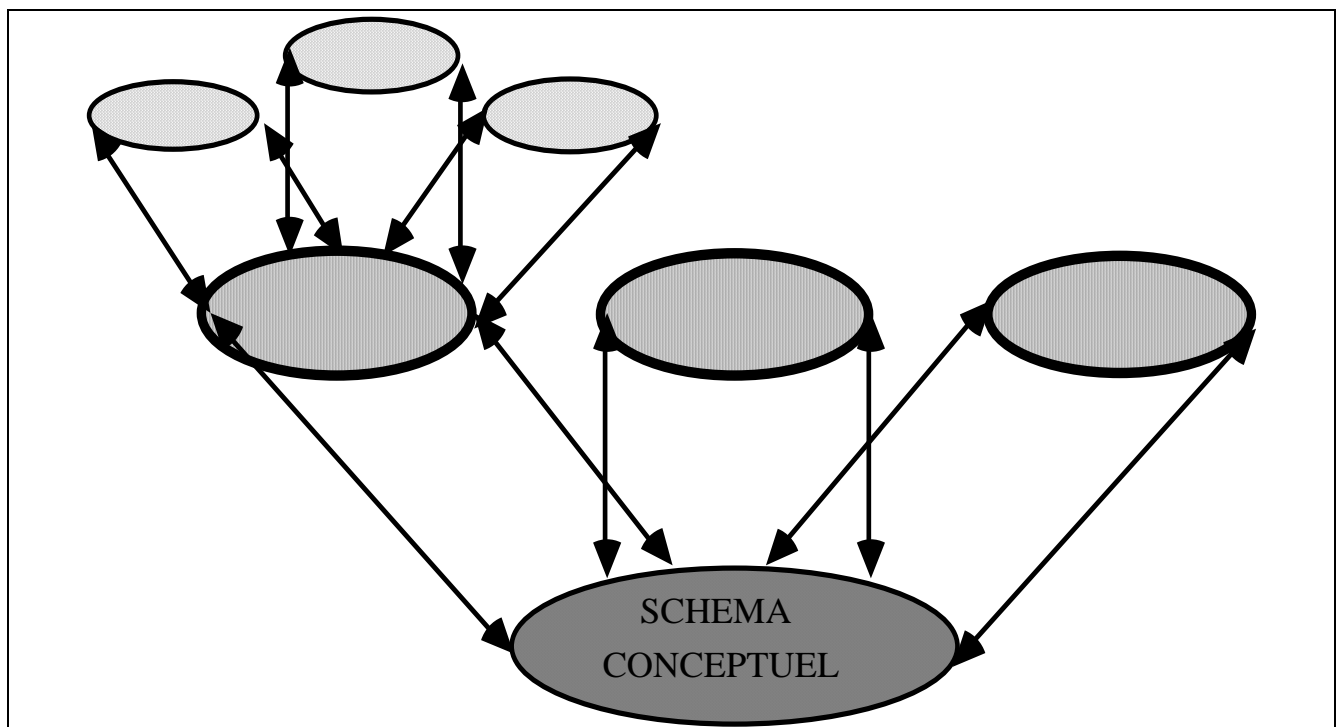
Décrit les parties des données présentant un intérêt pour un utilisateur ou un groupe d'utilisateurs.



SCHÉMA EXTERNE (VUE) :

Définit :

◇ Les sous-ensembles de données.



2.5 Historique des méthodes de conception de SI

2.5.1 Critères de classification des méthodes de conception de SI

La classification des méthodes de conception peut être faite selon les critères suivants :

◆ **Les étapes du cycle de vie** qu'elles supportent :

- ◇ Méthodes de conception
- ◇ Méthodes de développement
- ◇ Méthodes de test et de maintenance
- ◇ Méthodes de conduite de projets
- ◇ Etc.

◆ **La technologie** visée :

- ◇ Types de langages de programmation
- ◇ SGF
- ◇ Types de SGBD
- ◇ Types d'outils temps réel
- ◇ Etc.

◆ **Les types d'applications visées :**

- ◇ Applications de gestion
- ◇ Applications temps réel
- ◇ Application CAO
- ◇ Etc.

◆ **Type de perception du SI :**

- ◇ Point de vue fonctionnel
- ◇ Point de vue systémique
- ◇ Point de vue objet

◆ **Démarche de conception préconisée :**

- ◇ Décomposition hiérarchique
- ◇ Approche de composition ascendante

2.5.2 Classification des méthodes de conception de SI

Les deux critères de classification retenus sont :

- ◆ Le mode de perception du SI
- ◆ La démarche de conception

 Trois générations de méthodes de conception :

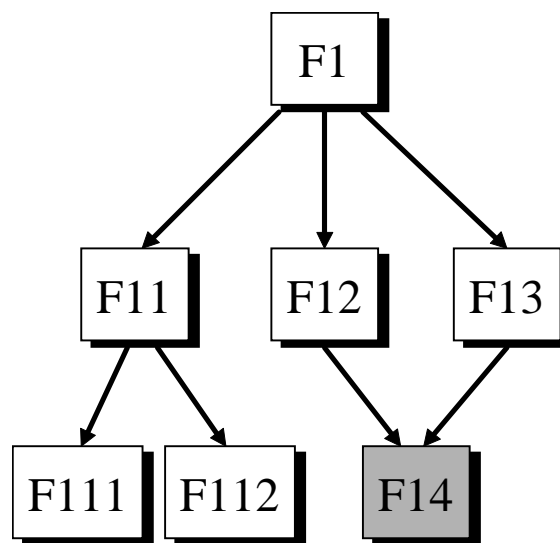
Génération	Période	Approche
1 ^{ère} génération	Années 70	<i>Méthodes analytiques ou cartésiennes</i>
2 ^{ème} génération	Années 80	<i>Méthodes systémiques</i>
3 ^{ème} génération	Années 90	<i>Méthodes orientés objet</i>

2.6 Méthodes analytiques ou cartésiennes¹

2.6.1 Principes

La démarche de ces méthodes consiste à

- ◆ **Découper** le domaine d'étude en fonctions.
- ◆ Prendre chaque fonction et la **décomposer** de façon **hiérarchique** en sous fonctions.
- ◆ Arrêter la décomposition lorsqu'on atteint un **niveau** de découpage **suffisamment fin** pour que le codage des sous fonctions soit simple à réaliser.



Ces méthodes privilégient l'approche par **traitements**.

¹ Dites aussi fonctionnelles.

2.6.2 Exemples

- ◆ Méthodes de **programmation structurée**
- ◆ Méthode **SADT** (Structured Analysis and Design Techniques).
- ◆ Méthode de **Jakson**
- ◆ Méthode de **Yourdon** (Modern Structured Analysis)

2.6.3 Avantages et inconvénients

◆ *Avantages :*

- ⇒ Correspond à la démarche naturelle pour aborder un problème.
- ⇒ Facilité de recenser les besoins des utilisateurs.
- ⇒ Facilité de produire des solutions à plusieurs niveaux d'abstraction.

◆ **Inconvénients :**

- ⇒ Concentration de l'effort d'analyse sur les traitements et négligence de la cohérence des données (redondance).
- ⇒ Absence de règles de décomposition produisant des hiérarchies de décompositions différentes selon les analystes.
- ⇒ Difficultés de tenir compte des interactions non hiérarchiques dans le cas de systèmes complexes.
- ⇒ L'intégration des différentes applications obtenues est peu conforme à la réalité que l'on a voulu décrire.



2.7 Méthodes systémiques

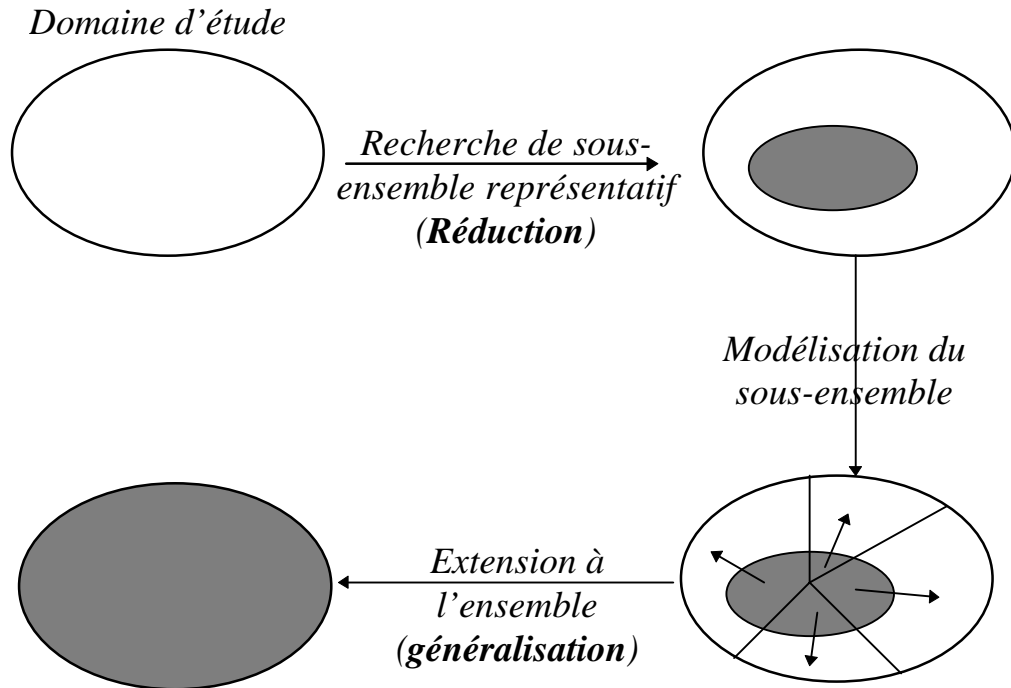
2.7.1 Principes

Ces méthodes sont basées sur les **concepts** suivants :

- ◆ Le SI est perçu comme un *objet complexe* actif dont il faut décrire la **structure** et les **objectifs fonctionnels**.
- ◆ La modélisation du SI est abordée selon deux points de vue complémentaires :
 - ◇ *Modélisation des données* : Aboutit à un modèle de données garantissant la cohérence des données.
 - ◇ *Modélisation des traitements* : Aboutit à l'élaboration d'un modèle de traitements décrivant les traitements à réaliser sur les données.

La **démarche** de ces méthodes est la suivante :

- ◆ Le domaine d'étude est représenté à l'aide d'un **modèle réduit** (sous-ensemble représentatif).
- ◆ Le modèle réduit est **découpé** en sous domaines.
- ◆ Chaque sous domaine est ensuite **étendu** à l'ensemble.



2.7.2 Exemples

- ◆ La méthode Merise
- ◆ La méthode Axial
- ◆ La méthode Information Engineering (IE)

2.7.3 Avantages et inconvénients

◆ *Avantages :*

- ⇒ Meilleure cohérence des données.
- ⇒ Respect des niveaux de représentation introduits par le groupe ANSI/SPARC (Niveau conceptuel, externe et interne).

◆ *Inconvénients :*

- ⇒ Absence de règles pour assurer la cohérence entre modèle des données et modèle des traitements.
- ⇒ Les frontières entre les niveaux conceptuel, interne et externe ne sont pas nettes.
- ⇒ Faiblesse de la modélisation des traitements : mélange des connaissances (règles de gestion) et du contrôle (contraintes d'intégrité).

2.8 Méthodes orientées objet

2.8.1 Principes

Les caractéristiques de ces méthodes sont les suivantes :

- ◆ Elles constituent une évolution des méthodes systémiques vers une **plus grande cohérence entre les objets et leur dynamique.**
- ◆ Basée sur le concept d'**objet.**
- ◆ Permet de décrire la **dynamique** du SI comme un ensemble d'**opérations rattachées aux objets** constituant le système.
- ◆ Cette représentation permet une meilleure modularité et réutilisation des composants du SI.
- ◆ C'est une approche ascendante :
 - ◇ Identification des objets de base du SI.
 - ◇ Par composition, constitution d'objets de plus en plus complexes.

2.8.2 Exemples

- ◆ **OOD** (G. Booch)
- ◆ **HOOD** (Hood Technical Group)
- ◆ **OOA** (S. Shlear et S. Mellor)
- ◆ **OOA / OOD** (T. Coal et E. Yourdon)
- ◆ **OMT** (J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen)
- ◆ **OOSE** (I. Jacobson, M. Cristerson, P. Jonson, G. Övergaard)
- ◆ **OOM** (M. Bouzeghoub et A. Rochfeld)

2.8.3 Avantages et inconvénients

◆ *Avantages :*

- ⇒ Grande capacité à modéliser les objets complexes.
- ⇒ Réduction des distorsions entre le réel et le système informatique.
- ⇒ Grande capacité à intégrer la dynamique des objets.
- ⇒ Possibilité d'encapsuler les parties privées.

◆ **Inconvénients :**

- ⇒ Risque d'avoir une perception monolithique des applications
- ⇒ Difficulté de l'effort d'abstraction



A retenir

- ◆ Étant donné l'augmentation du **niveau de complexité** des SI, du **nombre de participants**, de la **durée** de conception et de mise en œuvre, des **enjeux financiers** et des **risques**, l'utilisation d'une **méthode de conception** devient de plus en plus nécessaire.
- ◆ Les méthodes de conception proposent des **modèles**, une **démarche** et des moyens de **contrôle qualité** pour répondre à ces besoins.
- ◆ Il existe quatre **modèles de cycles de conception** et de développement de SI :
 - ◇ Modèle en **cascade**
 - ◇ Modèle en **V**
 - ◇ Modèle en **spirale**
 - ◇ Modèle **tridimensionnel**
- ◆ Selon l'architecture ANSI/SPARC, il existe trois niveaux de description des données :
 - ◇ Niveau **conceptuel**
 - ◇ Niveau **interne**
 - ◇ Niveau **externe**

◆ Trois générations de **méthodes de conception** de SI se sont succédées :

- ◇ Méthodes **analytiques** ou cartésienne (1^{ère} génération)
- ◇ Méthodes **systemiques** (2^{ème} génération)
- ◇ Méthodes **orientées objet** (3^{ème} génération)

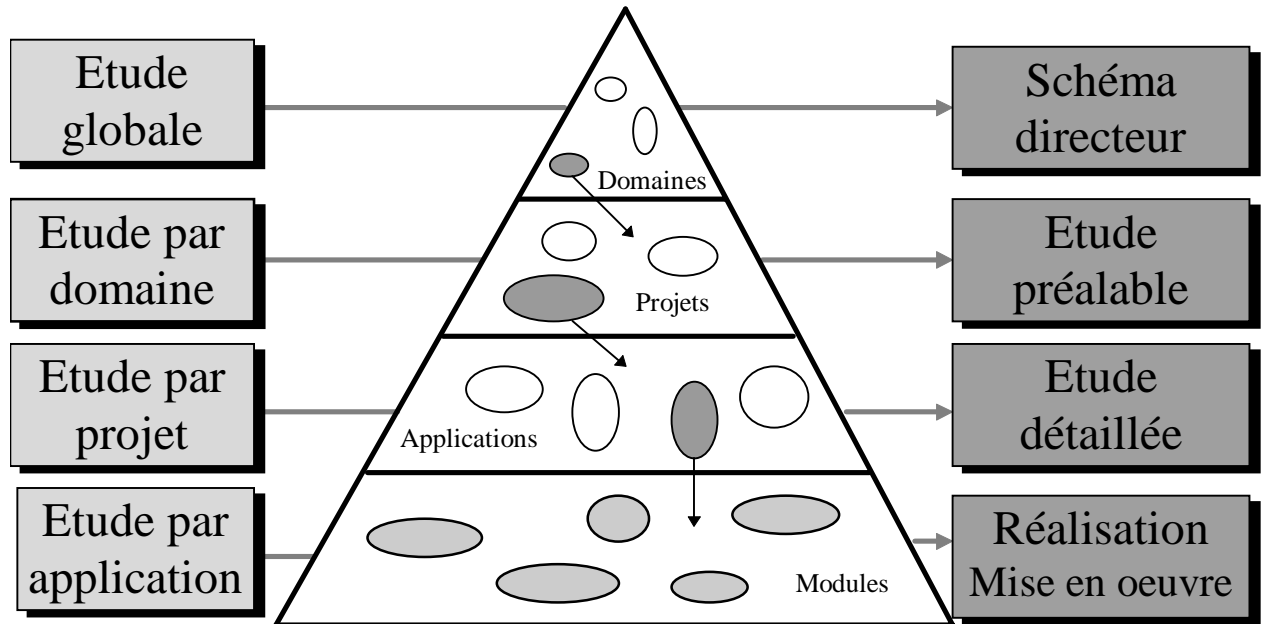
3. ÉTUDE D'UNE MÉTHODE SYSTÉMIQUE : MERISE

3.1 Introduction

- ◆ **Origine** : Fruit de la réflexion d'un groupe réuni par le ministère de l'industrie français au début des années 70.
- ◆ **Caractéristiques** :
 - ◇ Fait partie des méthodes *systemiques*.
 - ◇ Utilise le modèle tridimensionnel comme cycle de développement.
 - ◇ La modélisation du SI se fait selon deux axes :
 - ⇒ Modélisation des données
 - ⇒ Modélisation des traitements
- ◆ De moins en moins utilisée.
- ◆ Utilisée essentiellement pour la conception de bases de données.

3.2 Les étapes

Merise propose les étapes suivantes pour la conception et le développement du SI d'une entreprise :



3.2.1 Schéma directeur

Permet :

- ◆ L'élaboration d'une stratégie en matière de SI
- ◆ La définition des domaines d'étude
- ◆ La planification du développement de chaque domaine
- ◆ L'évaluation des moyens humains et financiers nécessaires pour chaque domaine

3.2.2 Étude préalable

- ◆ Porte sur un **sous-ensemble représentatif** du domaine étudié comportant les activités les plus importantes, les plus critiques et les plus sujettes au changement.
- ◆ Elle se fait selon trois phases :
 - ◇ Recueil
 - ◇ Conception
 - ◇ Appréciation

◆ **Phase de recueil** : Elle consiste à recadrer la mission et étudier l'existant.

◇ *Définition de la mission* : Il s'agit de définir le **champ d'étude**, de préciser les **objectifs**, de **planifier** les grandes lignes et constituer les **groupes de travail**.

⇒ Validation : Accord des décideurs sur les objectifs.

◇ *Diagnostic de l'existant* : Il s'agit d'acquérir une bonne **connaissance du domaine étudié**, d'identifier un **sous-ensemble représentatif**, de **décrire** ce sous ensemble représentatif et d'effectuer un **bilan de l'existant**.

⇒ Entretiens avec les acteurs

⇒ Représentation des flux à l'aide de diagramme de flux :

⇒ Représentation de l'existant à l'aide des modèles Merise (MCT, MCD, ...).

⇒ Validation : Accord des décideurs des acteurs sur le SER et sur la description de l'existant.

◆ **Phase de conception** : Elle consiste à définir des solutions organisationnelles et techniques. Se fait selon trois étapes:

◇ *Définition des orientations* : Il s'agit de choisir et fixer des orientations en matière de gestion, d'organisation et technique.

⇒ Validation : Accord des décideurs (et experts) sur la validité des orientations.

◇ *Conception (générale) des solutions* : Il s'agit d'élaborer :

- une description conceptuelle des traitements
- une description organisationnelle des traitements en fonction des solutions d'organisation
- un fonctionnement en mode dégradé
- une description conceptuelle des données
- un dimensionnement des solutions

⇒ Validation : Accord des décideurs et des utilisateurs sur les solutions proposées.

◇ **Recherche des solutions techniques** : Il s'agit d'élaborer des solutions techniques correspondant aux solutions conceptuelles et organisationnelles :

- Répartition des données et des traitements
- Élaboration d'architecture matérielle et logicielle

◆ **Phase d'appréciation** : Elle consiste à déterminer des scénarios de réalisation des solutions proposées et d'en quantifier les coûts, les services rendus, les risques et les enjeux. Elle se fait selon les étapes suivantes :

◇ **Élaboration de scénarios de mise en oeuvre** : Il s'agit d'effectuer un découpage en projets, de définir les étapes, les modalités de recette et un calendrier général.

◇ **Évaluation** : Il s'agit d'évaluer :

- le coût matériel et logiciel
- la mise en œuvre et l'organisation
- les systèmes transitoires
- les risques et les enjeux

◇ **Bilan et choix** : Il s'agit d'effectuer un bilan comparatif, de rédiger un dossier de choix, le soumettre et effectuer un choix de solution :

⇒ Validation : Accord sur les choix effectués.

3.2.3 Étude détaillée

- ◆ Consiste à généraliser et à détailler le travail effectué pendant la phase d'étude préalable. Elle complète les descriptions effectuées lors de l'étude préalable et respecte les solutions décidées à l'issue de cette étude.

- ◆ Elle se fait selon deux phases :

 - ◇ Conception générale

 - ◇ Conception détaillée

- ◆ **Phase de conception générale** : Elle consiste à affiner les descriptions conceptuelles et organisationnelles des traitements. Elle aboutit à l'élaboration d'un « dossier de spécifications générales ». Elle se fait en deux étapes :

 - ◇ **Inventaire** : Il s'agit de définir la **typologie des événements et des résultats** et d'affiner la description conceptuelle des traitements.

 - ⇒ Validation : Accord sur le détail des règles de gestion.

 - ◇ **Découpage en procédures** : Il s'agit de définir les procédures de traitement et les postes de travail afin de définir une description organisationnelle des traitements.

 - ⇒ Validation : Accord sur le dossier de spécifications générales.

◆ **Phase de conception détaillée** : Elle consiste à analyser d'une façon détaillée chaque procédure, compléter la description conceptuelle des données et détailler et valider les choix techniques. Elle aboutit à l'élaboration d'un « dossier de spécifications détaillées ». Elle se fait en quatre étapes :

◇ *Analyse détaillée de chaque procédure* : Consiste à étudier d'une façon détaillée les procédures conversationnelles et les postes de travail associés ainsi que les procédures de traitement différées.

⇒ Validation : Accord sur la description détaillée de chaque procédure.

◇ *Compléter la description conceptuelle des données* : Il s'agit de compléter et de valider la description conceptuelle des données.

⇒ Validation : Accord sur la description conceptuelle des données.

◇ *Compléter et valider les choix techniques* : Il s'agit de compléter et de valider les choix techniques faits dans l'étude préalable.

⇒ Validation : Accord sur les choix techniques.

◇ *Planification de la réalisation* : Il s'agit de planifier et de préparer la phase de réalisation.

3.2.4 Réalisation

- ◆ Consiste à produire les programmes selon les spécifications de l'étude détaillée avec test et validation.

- ◆ Elle se fait selon deux phases :

 - ◇ Étude technique

 - ◇ Production des programmes

- ◆ **Phase d'étude technique** : Elle consiste à aboutir à une description physique des données et des traitements. Elle se fait en deux étapes :

 - ◇ **Organisation physique des données** : Il s'agit de définir un modèle logique et un modèle physique des données.

⇒ Validation : Validation de l'organisation et du volume.

 - ◇ **Organisation physique des traitements** : Il s'agit de définir l'architecture technique des programmes transactionnels et programmes différés (dessin d'écran et états, prototypes).

⇒ Validation : Accord sur le cette organisation technique.

◆ **Phase de développement** : Elle consiste à développer les programmes et les mettre au point. Elle se fait en trois étapes :

◇ ***Planning de production*** : Consiste à planifier la production des programmes et de réception interne.

⇒ Validation : Accord sur le planning et les modalités de réception.

◇ ***Développement*** : Il s'agit de produire les programmes et de les mettre au point en interne.

⇒ Validation : Tes unitaires et validation interne.

◇ ***Mise au point globale*** : Il s'agit d'effectuer des tests d'intégration.

⇒ Validation : Réception par le client.


3.3 La modélisation

La modélisation dans Merise consiste à représenter le système d'information selon trois niveaux d'abstraction:

- ◆ **Niveau conceptuel** : Représente la finalité du système en s'appuyant sur ses objectifs et ses aspects invariants indépendamment de la manière dont ils seront réalisés.

 *Le QUOI*

- ◆ **Niveau logique ou organisationnel** : Définit l'organisation que l'on doit mettre en place pour atteindre les objectifs décrits au niveau conceptuel.

 *Le QUI, le OÙ et le QUAND*

- ◆ **Niveau physique ou opérationnel** : Décrit l'implémentation du système d'information en tenant compte des choix techniques arrêtés.

 *Le COMMENT*

Cette modélisation est faite pour :

- ◆ les données
- ◆ les traitements



Six modèles de base

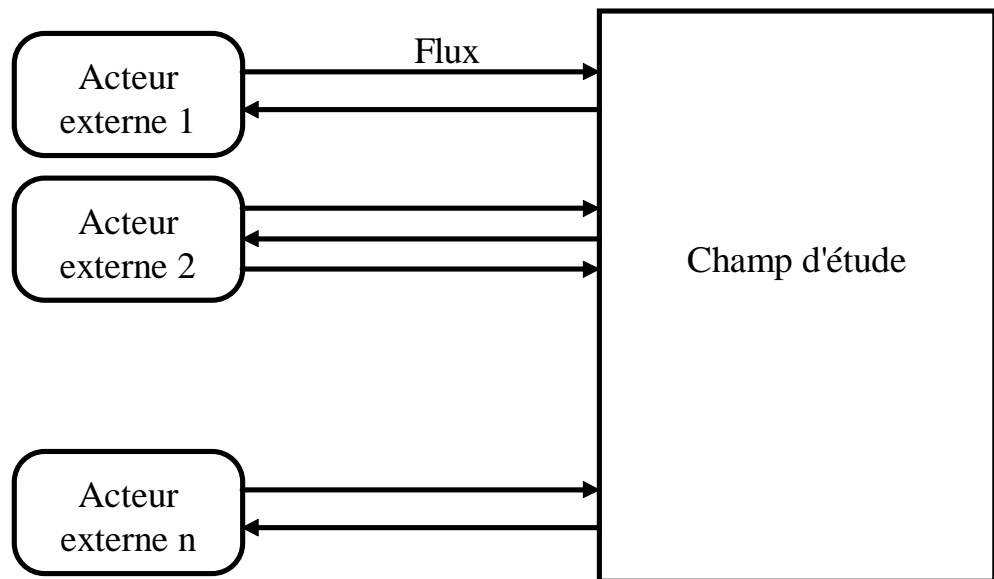
	Données	Traitements
<i>Niveau conceptuel</i>	Modèle conceptuel de données (MCD)	Modèle conceptuel de traitements (MCT)
<i>Niveau logique / organisationnel</i>	Modèle logique de données (MLD)	Modèle organisationnel de traitements (MOT)
<i>Niveau physique / opérationnel</i>	Modèle physique des données (MPD)	Modèle opérationnel des traitements (MOP)

Autres modèles :

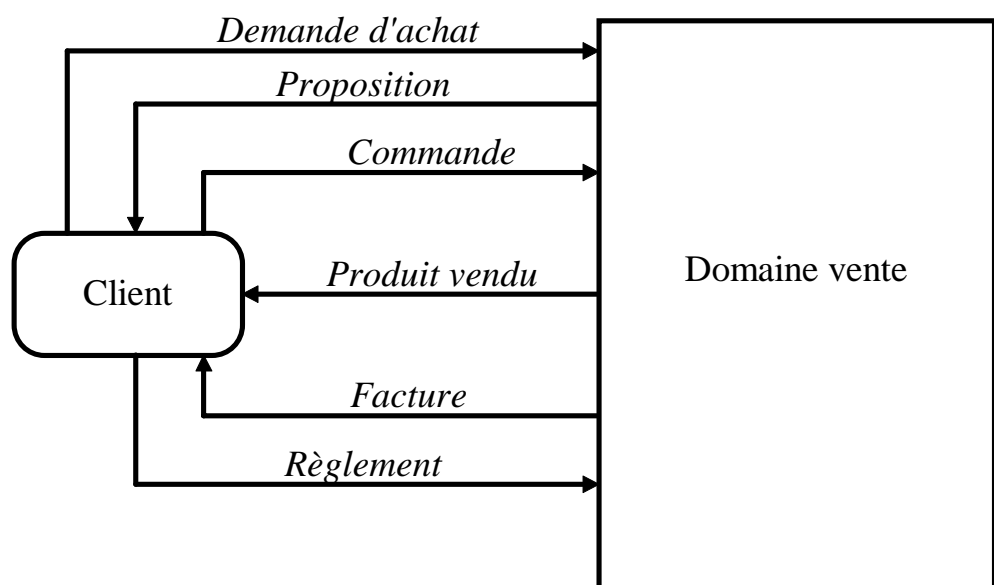
Deux modèles supplémentaires ont été rajoutés dans le cadre de l'extension de la méthode :

◇ **Modèle de contexte** : Permet de décrire les échanges:

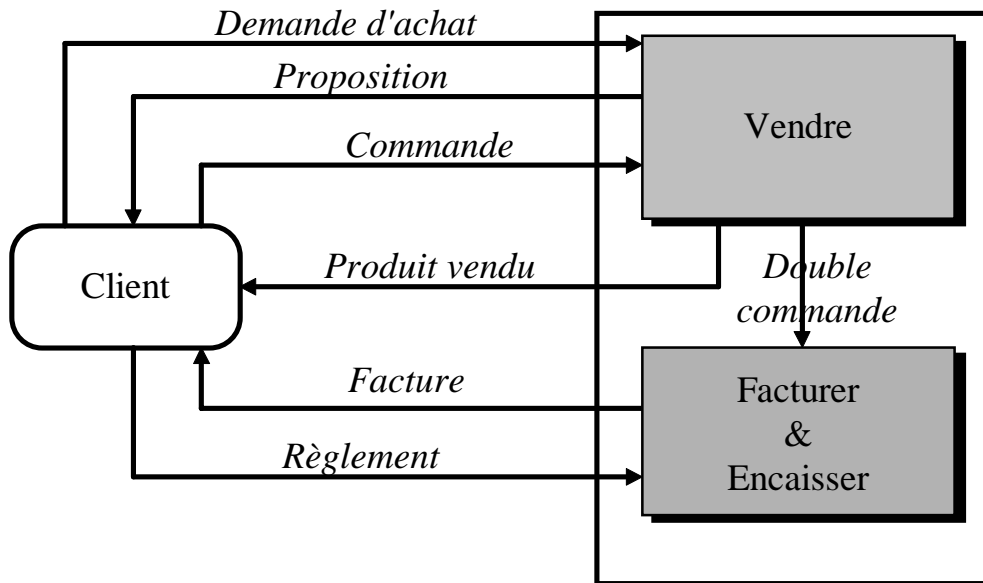
- entre le système global et son environnement



- entre un domaine et l'environnement externe



◇ **Modèle de flux conceptuel** : C'est une extension du modèle de contexte. En plus des échanges avec l'environnement externe d'un domaine, le MFC décrit l'activité à l'intérieur du domaine. Les MFD sont obtenus par décompositions successives jusqu'à atteindre un niveau suffisant de découpage.



Remarque :

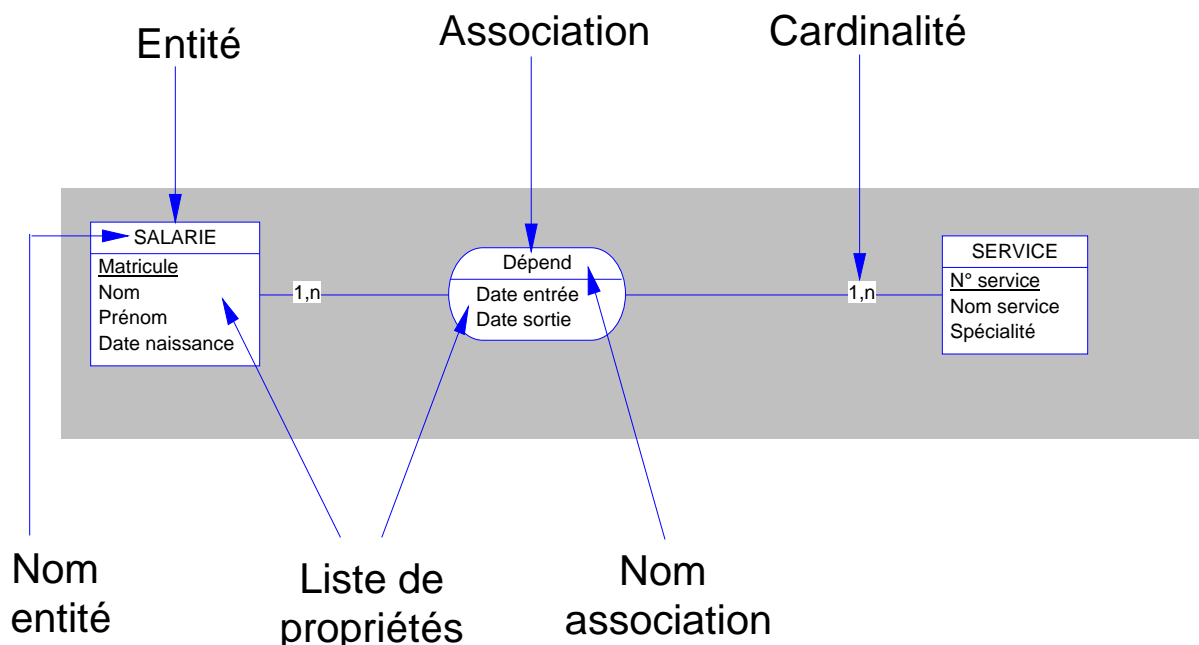
On parle aussi de modèle conceptuel de communication (MCC) qui permet d'effectuer les deux représentations précédentes.

3.4 Modélisation des données

3.4.1 Modèle conceptuel de données

Présentation :

- ◆ Permet d'effectuer une représentation **conceptuelle** de l'ensemble de données manipulées et des **règles de gestion** auxquelles elles sont soumises.
- ◆ Décrit la **sémantique** des données indépendamment de leur utilisation et de leur implémentation (Niveau conceptuel de l'architecture ANSI/SPARC/X3).
- ◆ Donne une représentation **statique** du système d'information de l'entreprise.



Terminologie :

◆ Propriété :

Information élémentaire représentant la plus petite partie manipulée dans d'entreprise et ayant un sens.

Exemples : matricule, nom, poids, prix, etc..

Règles :

- ◇ Une propriété peut être **simple** (*ex* : *salaire, prix, etc..*) ou **composée** (*ex* : *adresse, date, etc..*) mais elle doit être toujours **atomique**.
- ◇ Une propriété ne doit pas être **calculée** (*ex* : *prix TTC, durée, etc..*).
- ◇ Une propriété ne doit jamais être **redondante** dans le MCD :
 - ⇒ Interdiction des **synonymes** (*ex* : *référence article et N° produit*) et des **polysèmes** : même signifiant pour plusieurs signifiés (*ex* : *"adresse" qui désigne "adresse client" et "adresse fournisseur"*).

◆ **Entité (ou objet) :**

C'est une association de propriétés correspondant à un type d'objet ayant un intérêt pour l'entreprise.

Exemples : article, employé, etc.

Règles :

- ◇ Parmi les propriétés d'une entité, il existe un sous-ensemble qui joue le rôle **d'identifiant** (*ex : référence article, matricule employé, etc..*).
- ◇ Un identifiant permet de connaître sans aucune ambiguïté toutes occurrences.
- ◇ Le choix d'identifiant se fait en appliquant la règle de **dépendance fonctionnelle monovaluée** :

$$A \rightarrow B$$

(ou A détermine B)

si la connaissance de A permet d'identifier B sans aucune ambiguïté.

Exemples :

N° matricule → Nom salarié

N° matricule → Prénom salarié

Nom salarié ~~→~~ Prénom salarié

- ◇ La **construction** d'une entité peut se faire en déterminant la (les) propriété(s)

qui joue(nt) le rôle d'identifiant puis en appliquant les dépendances fonctionnelles, trouver les autres propriétés.

◆ Association (ou relation) :

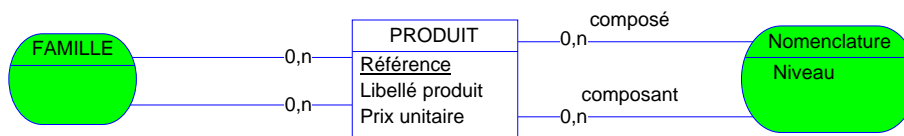
C'est un mécanisme permettant de représenter les liens entre deux ou plusieurs entités. Il traduit le langage de l'entreprise.

Exemples : une commande "CONCERNE" des articles, un service "COMPREND" des employés, etc.

Règles :

- ◇ L'ensemble d'entités intervenant dans une association constitue une **collection**.
- ◇ Une association peut être **porteuse** ou non de propriétés.
- ◇ La **dimension** de l'association est le nombre d'entités entrant dans sa collection. Elle peut être binaire, ternaire, etc. ou de dimension n.
- ◇ Les **propriétés des associations** sont des propriétés qui sont en dépendance fonctionnelle de deux ou plusieurs identifiants d'entités (*ex* : *quantité commandée*).

- ◇ **L'identifiant** d'une association est la combinaison des identifiants des entités associées. Cet identifiant est implicite, il n'est pas représenté sur le MCD.
- ◇ Une **association réflexive** est une association qui relie une entité à elle-même (*ex : lien de parenté entre personnes*).
- ◇ Une association réflexive peut être **symétrique** ou **orientée**.



◆ Cardinalités :

Représente pour chaque couple (entité, association) les nombres minimum et maximum d'occurrences de l'association que peut avoir un objet.

Exemples : un service "COMPREND" un ou plusieurs employés.

Règles :

- ◇ Les seules valeurs possibles sont : (0, 1), (0, n), (1, 1) et (1, n).
- ◇ Les cardinalités traduisent les règles de gestion

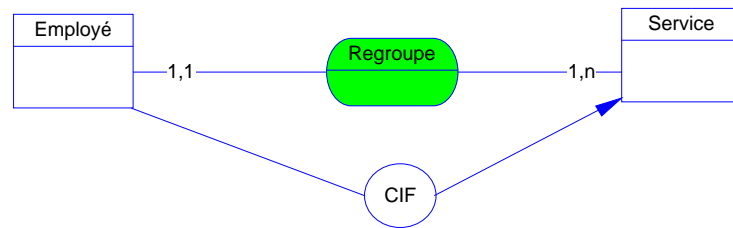
◆ **Contraintes d'intégrité fonctionnelle (CIF) :**

◇ Une CIF définie sur une association permet de représenter le fait que l'une des entités de sa collection est identifiée sans ambiguïté par la connaissance d'une ou plusieurs autres.

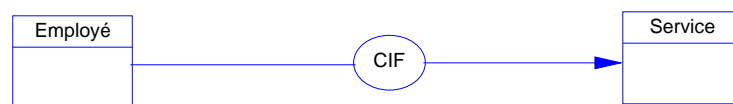
Entité 1 → Entité 2

◇ Les associations binaires ayant les cardinalités (0,1) ou (1,1) constituent une CIF. Si l'association n'est pas porteuse de propriétés, elle peut être remplacée par la CIF.

Exemple :

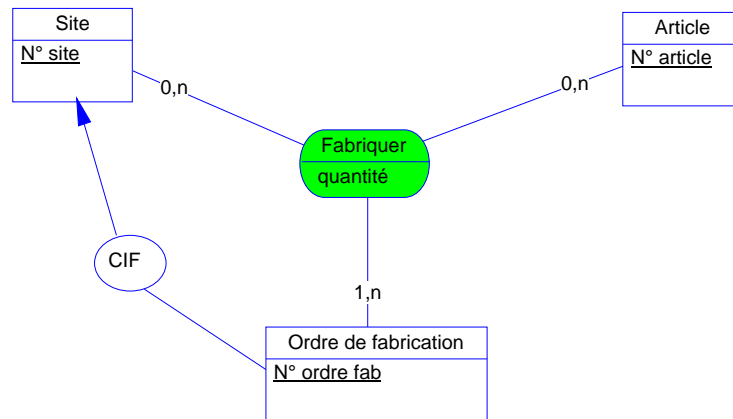


Peut être remplacée par :



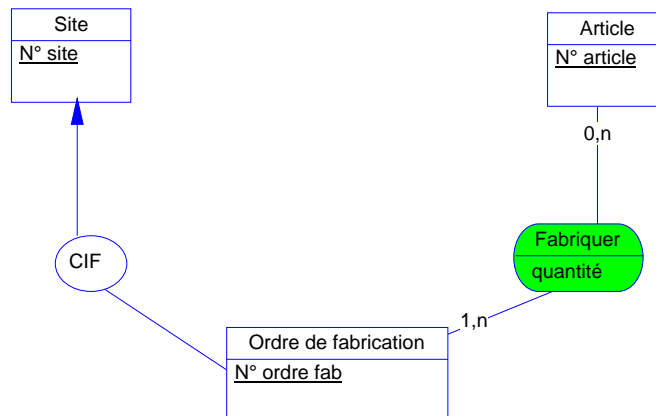
◇ Les CIF permettent de simplifier les associations de dimension supérieure à 2.

Exemple : Soit le modèle suivant :



avec la contrainte suivante : "un ordre de fabrication ne concerne qu'un seul site".

⇒ Simplification du modèle :



Exemples : une commande "CONCERNE" des articles, un service "COMPREND" des employés, etc.

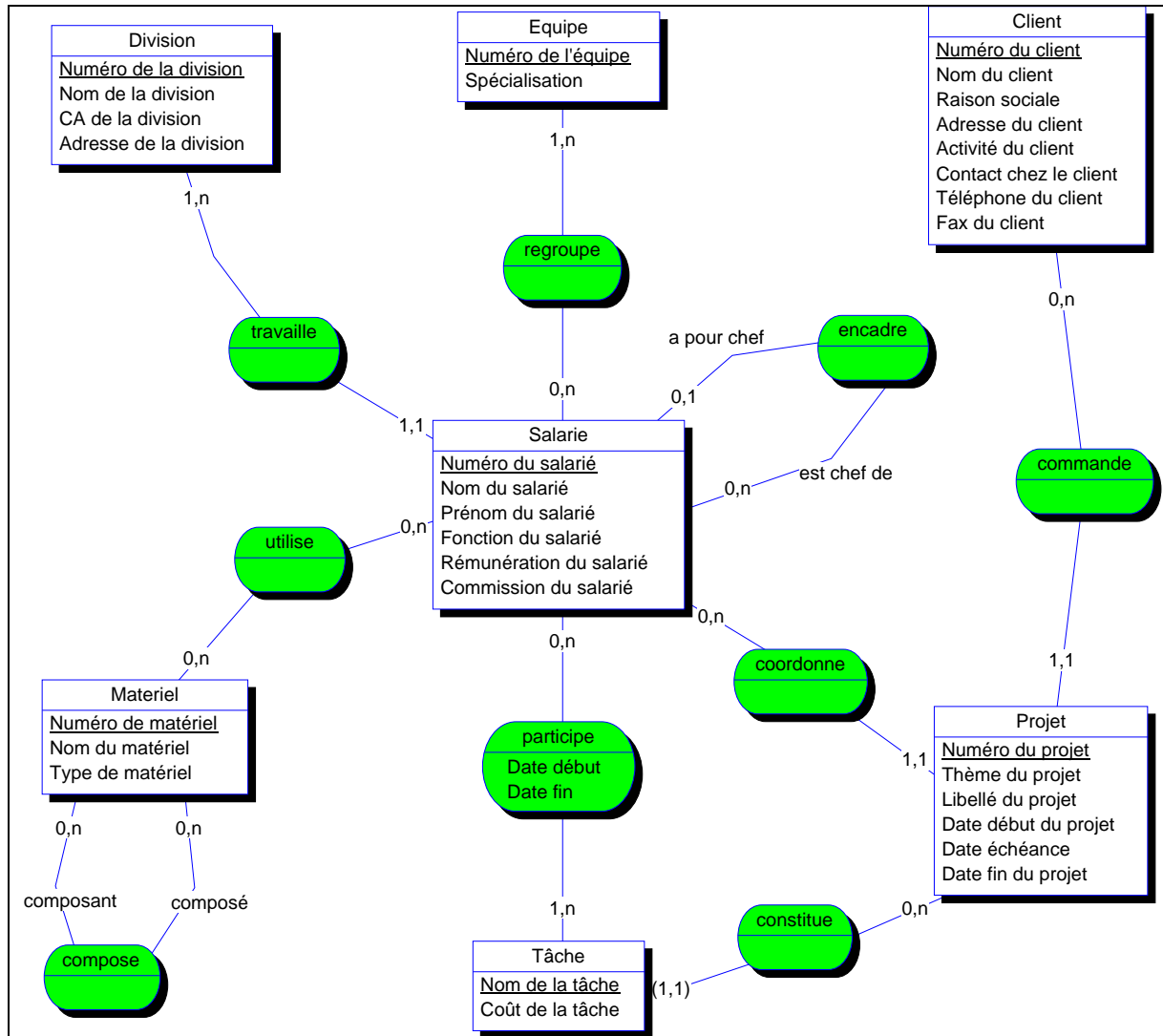
Règles :

- ◇ Une association peut être **porteuse** ou non de propriétés.
- ◇ La **dimension** de l'association est le nombre d'entités entrant dans sa

collection. Elle peut être binaire, ternaire, etc.. ou de dimension n.

- ◇ Les **propriétés des associations** sont des propriétés qui sont en dépendance fonctionnelle de deux ou plusieurs identifiants d'entités (ex : quantité commandée).

Exemple de modèle :



Vérification d'un MCD :

L'application des règles suivantes à un MCD permet de s'assurer que le modèle physique qu'on va obtenir correspond bien à la réalité.

Règle 1 :

Toutes les propriétés doivent être élémentaires, c'est à dire non décomposables.

→ Décomposer les propriétés composées.

Exemple : adresse

Règle 2 :

Chaque entité doit avoir un identifiant et un seul.

→ En cas d'existence de plusieurs, à choisir un en fonction de l'utilisation.

Exemple : N° matricule et N° d'identification national.

Règle 3 :

Pour une entité, les propriétés autres que l'identifiant doivent être en dépendance fonctionnelle monovaluée avec cet identifiant.

→ Décomposition de l'entité

Exemple : N° matricule → → Diplôme

Règle 4 :

Une propriété ne peut qualifier qu'une seule entité ou qu'une seule association.

→ Éliminer les redondances, les synonymes et les polysèmes.

Exemple : nom, adresse.

Règle 5 :

Il ne doit pas y avoir de dépendances fonctionnelles transitives dans une entité : dépendances fonctionnelles entre les propriétés non identifiantes.

→ Éclater l'entité

Exemple : **Client** (N° client, nom client, ... catégorie client, taux remise)

avec

N° client → catégorie client → taux remise.

Donc :

Client (N° client, nom client, ...)

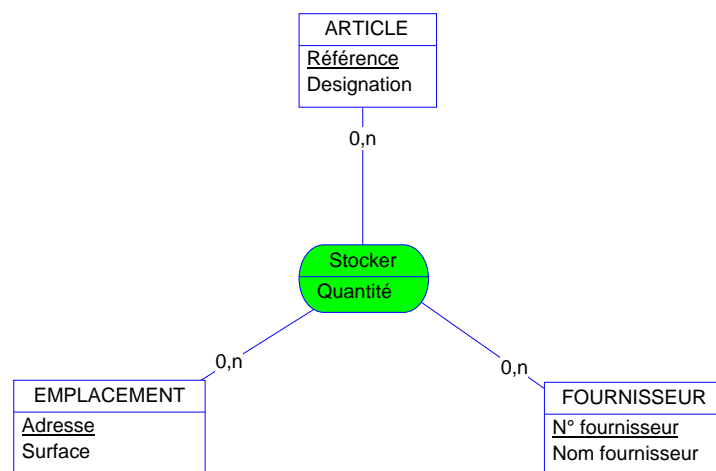
Catégorie (Code catégorie client, intitulé catégorie, taux remise)

Règle 6 :

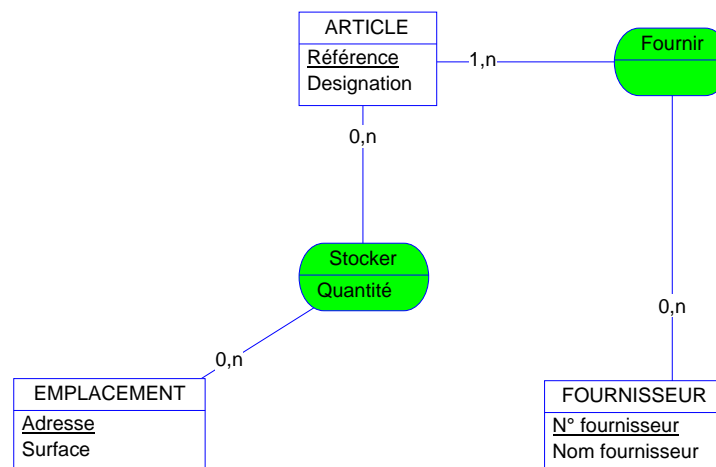
Pour chaque occurrence d'une association, il doit exister une et une seule occurrence de chacune des entités de la collection.

→ Éclater l'association.

Exemple : Un article est fourni par un fournisseur et stocké dans un emplacement.



à remplacer par



Règle 7 :

Les propriétés d'une association doivent dépendre de la totalité de l'identifiant de cette association.

→ Éclater l'association.

Démarche pour la construction d'un MCD :

Deux approches :

- ◇ Approche pragmatique
- ◇ Approche formelle

◆ **Approche pragmatique** : Se base sur l'intuition du concepteur. Les entités et les associations sont construites à partir des définitions de l'existant.

Avantage : rapidité

Inconvénients : risque de ne modéliser que l'existant et non pas quelque chose d'invariant.

◆ **Approche formelle** : consiste à suivre une démarche rigoureuse :

- 1) Établir une liste des propriétés
- 2) Classer cette liste par ordre alphabétique
- 3) Éliminer les redondances, les synonymes et les polysèmes
- 4) Repérer les identifiants existants et dégager les entités naturelles

- 5) Rattacher à ces entités les propriétés en dépendance fonctionnelle avec leurs identifiants.
- 6) Placer les associations et leur rattacher les propriétés en dépendance fonctionnelles avec plusieurs identifiants.
- 7) Revoir les propriétés restantes afin de les regrouper dans des entités pour lesquelles seront créés des nouveaux identifiants.
- 8) Étudier les cardinalités
- 9) Simplifier le modèle à l'aide de CIF
- 10) Vérifier le modèle à l'aide des règles.

3.4.2 Modèle logique de données (MLD)

Présentation :

- ◆ Le MLD constitue une étape intermédiaire entre le modèle conceptuel et le modèle physique de données.
- ◆ C'est le **MCD** auquel on rajoute la définition de l'**organisation logique** des données et en l'**optimisant** compte tenu des traitements à appliquer aux données.
- ◆ A ce niveau, on doit choisir le mode d'organisation des données :
 - ◇ modèle hiérarchique
 - ◇ modèle réseau
 - ◇ **modèle relationnel**
 - ◇ fichiers classiques

Concepts structurels du relationnel :

- ◆ Domaines
- ◆ Attributs
- ◆ Relations
- ◆ Schéma d'une relation
- ◆ Tuples
- ◆ Clé primaire
- ◆ Clé étrangère
- ◆ Clé candidate
- ◆ Formes normales

Règles de passage :

Les règles suivantes sont appliquées pour passer d'un MCD à un MLD relationnel :

Règle 1 :

Entité → Relation
Propriété → Attribut
Identifiant → Clé primaire

Règle 2 :

Une association binaire ayant des cardinalités $(x, 1)$ et (x, n) , x étant égale à 0 ou 1, se traduit par :

- ◇ la migration de l'identifiant de l'entité ayant la cardinalité (x, n) vers l'entité ayant la cardinalité $(x, 1)$
- ◇ Cet identifiant devient une clé étrangère.
- ◇ Dans le cas d'association reflexive, l'identifiant est dupliqué puis renommé.
- ◇ la migration des propriétés de l'association (s'il y en a) vers l'entité de cardinalité $(0, 1)$.

Règle 3 :

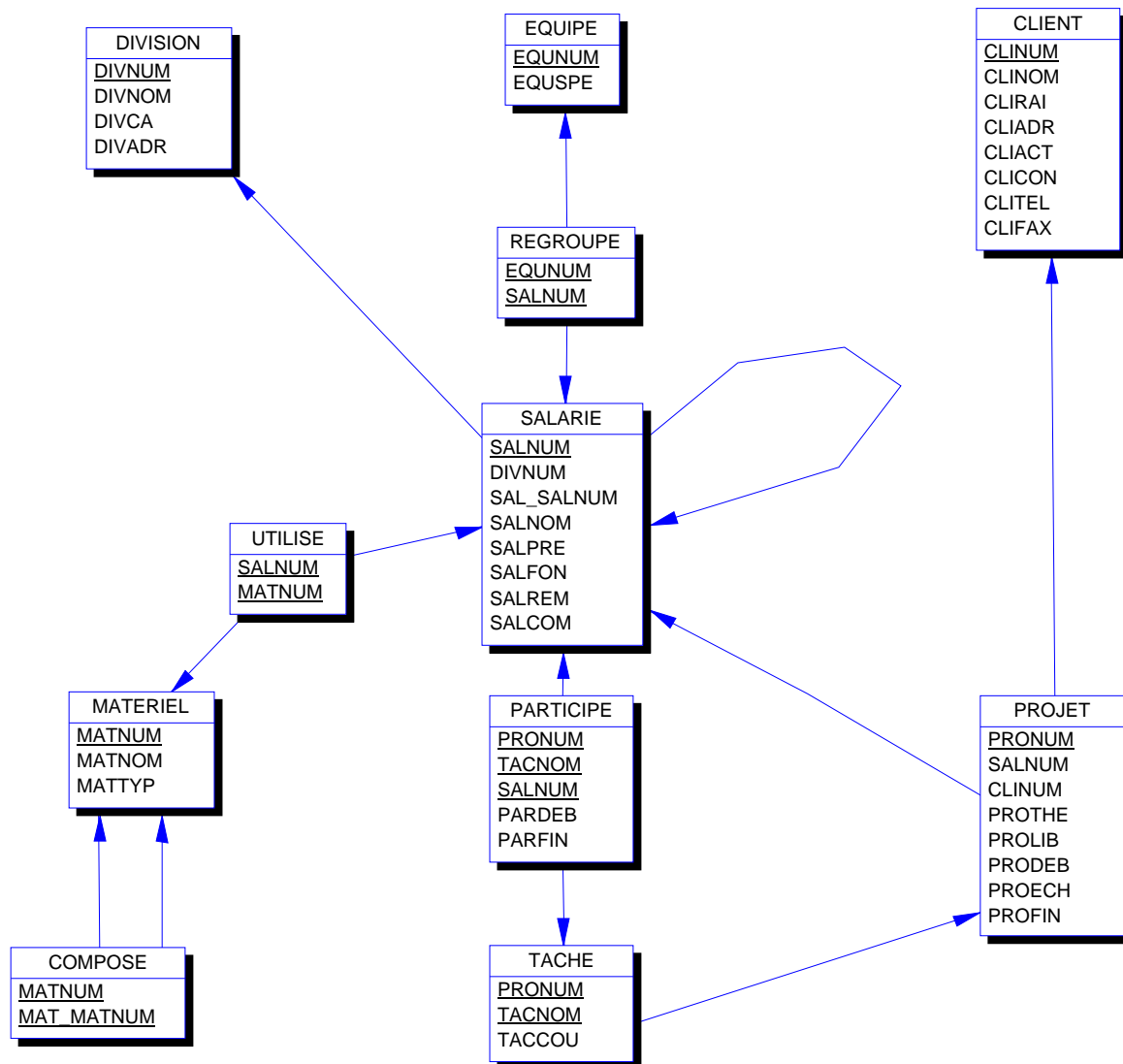
Une association binaire ayant des cardinalités (x, n) et (x, n) , x étant égale à 0 ou 1, se traduit par :

- ◇ la création d'une nouvelle relation
- ◇ la migration de l'identifiant de chacune des entités vers la nouvelle relation. L'ensemble de ces identifiants constitue la clé primaire.
- ◇ Chacun des identifiants devient une clé étrangère.
- ◇ Les propriétés de l'association constituent le reste des attribues de la nouvelle table.
- ◇ Dans le cas d'association réflexive, l'identifiant est dupliqué puis renommé.

Règle 4 :

- ◇ Une association n-aire ($n > 2$) porteuse ou non de propriétés, se transforme en une relation ayant comme clé primaire la composition de l'ensemble des identifiants de la collection et comme attributs ceux de l'association.

Exemple de MLD :



3.4.3 Modèle physique de données

Présentation :

- ◆ Le MPD permet d'implémenter le MLD selon les spécificités du SGBD utilisé. Ces spécificités concernent :
 - ◇ les règles de nommage
 - ◇ les types de données
 - ◇ l'expression des contraintes d'intégrité
 - ◇ les types d'accélérateurs (index, clusters, ...)
 - ◇ les paramètres de stockage
- ◆ Un MPD se présente sous forme d'un ou plusieurs scripts contenant des commandes (SQL) de création de structures de données.
- ◆ Généré automatiquement par la plupart des AGL.

Exemple de MPD (Oracle)

```
--
=====
--   Table : SALARIE
--
=====
create table SALARIE
(SALNUM      NUMBER(4)  not null
   constraint ck_salarie_salnum
      check (SALNUM between 1 and 9999),
DIVNUM      NUMBER(4)  not null
   constraint ck_salarie_divnum
      check (DIVNUM between 1 and 9999),
SAL_SALNUM  NUMBER(4)
   constraint ck_salarie_sal_salnum
      check (SAL_SALNUM between 1 and 9999),
SALNOM      VARCHAR2(40)
SALPRE      VARCHAR2 (40)
SALFON      VARCHAR2 (40)
SALREM      NUMBER(8,2)
   constraint ck_salarie_salrem check (SALREM >= 0),
SALCOM      NUMBER(8,2)
   constraint ck_salarie_salcom check (SALCOM >= 0),
   constraint pk_salarie primary key (SALNUM));
--
=====
--   Index : SALARIE_FK1
--
=====
create index SALARIE_FK1 on SALARIE (DIVNUM asc);
```

Voir Annexe A.

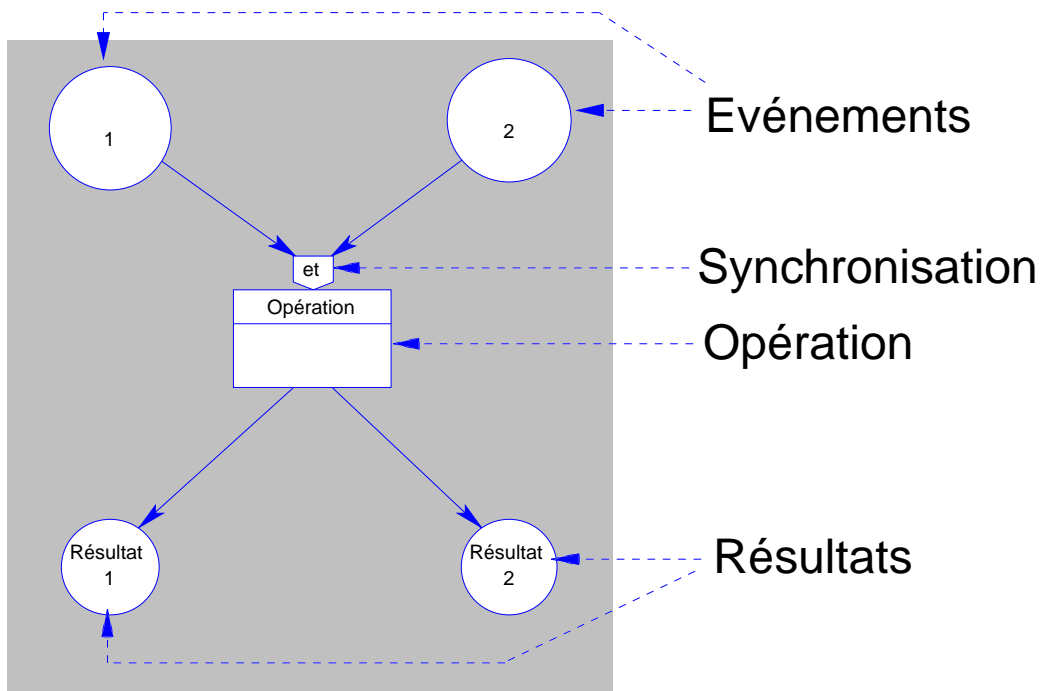
3.5 Modélisation des traitements

3.5.1 Modèle conceptuel de traitements

Présentation :

- ◆ Permet d'effectuer une représentation **conceptuelle** des traitements effectués dans l'entreprise.
- ◆ En s'appuyant sur la spécification des règles de gestion, il représente
 - ◇ les événements,
 - ◇ les résultats,
 - ◇ les opérations et
 - ◇ les synchronisations.
- ◆ Donne une représentation **dynamique** du système d'information de l'entreprise.

Formalisme :



Terminologie :

◆ Événement :

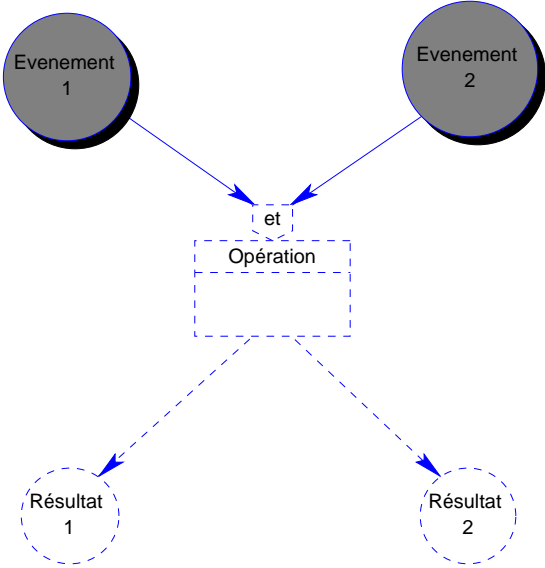
C'est la représentation d'un fait nouveau porteur d'information pour le système étudié.

Il peut représenter un non événement.

Exemples : réception commande, absence de réponse

Règles :

- ◇ Les seuls événements pris en compte dans le MCT sont les **événements externes** (indépendants de l'organisation).



◆ Opération :

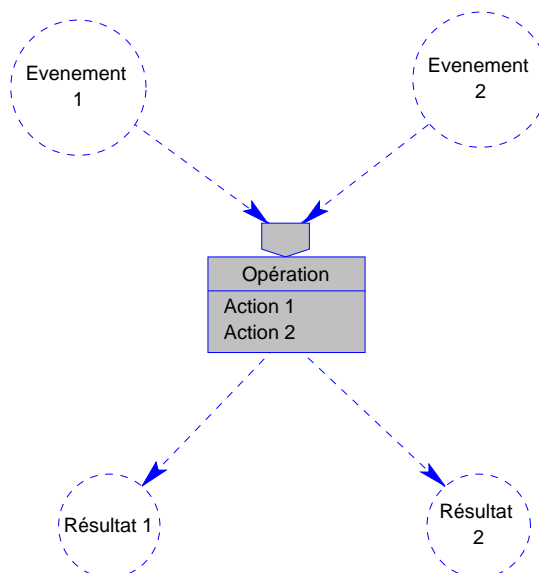
C'est la réaction du système, sous forme de traitement, à l'arrivée d'un ensemble d'événements.

Une opération est composée d'un ensemble **d'actions**.

Exemples : traitement de commande, paiement facture

Règles :

- ◇ L'opération traite et prend en compte les informations en provenance des événements.
- ◇ Une opération déclenchée ne doit pas se trouver en attente d'un autre événement.



◆ Synchronisation :

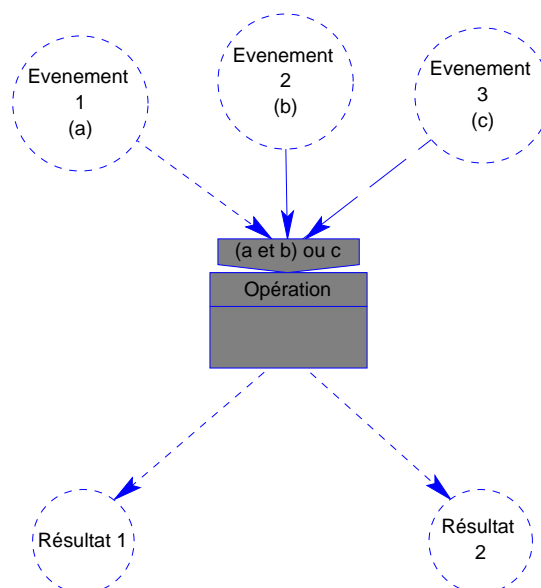
C'est l'association de deux ou plusieurs événements pour le déclenchement d'une opération.

C'est une expression booléenne formée à partir des opérateurs ET et OU.

Exemples : Facture reçue **et** marchandise réceptionnée

Règles :

- ◇ L'opérateur ET est pris par défaut.
- ◇ Pour une meilleure lisibilité, il est recommandé d'attacher à chaque événement un label.
- ◇ La présence des événements n'entraîne pas le déclenchement de l'opération. C'est la description organisationnelle qui précisera ce moment.



◆ **Résultat :**

C'est la réponse du système aux événements ayant déclenché une opération.

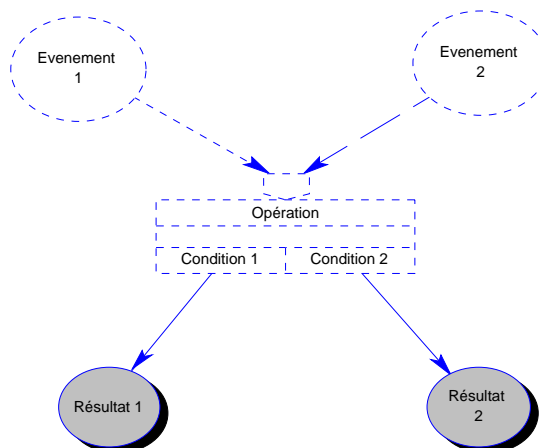
Exemples : commande traitée, marchandise livrée

Règles :

- ◇ Une opération peut produire plusieurs résultats.
- ◇ La production des résultats peut être soumise à des conditions de sortie de l'opération.
- ◇ Le résultat d'une opération peut participer en tant qu'événement dans une autre opération.

→ Événement interne

- ◇ Une opération ne peut pas être déclenchée que par des événements internes. Il doit y avoir au moins un événement externe.



◆ **Processus :**

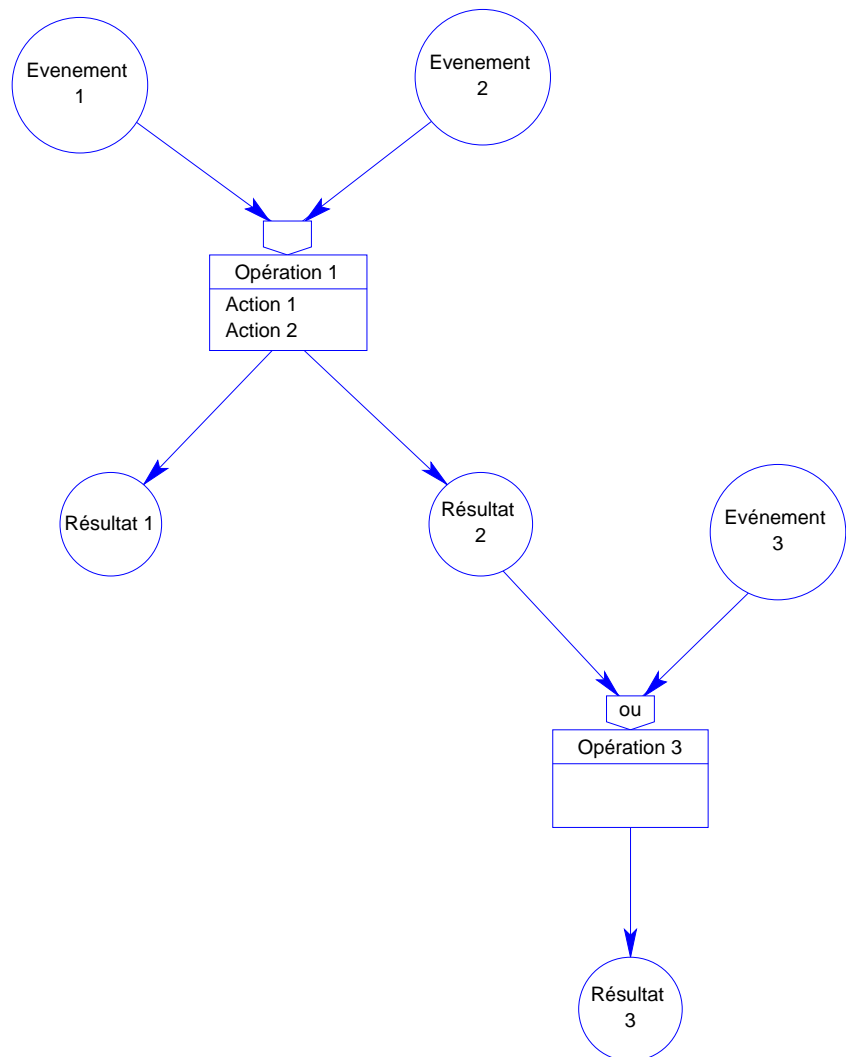
C'est un enchaînement synchronisé d'opérations représentant une unité homogène de traitement.

Un processus est propre à un domaine d'activité.

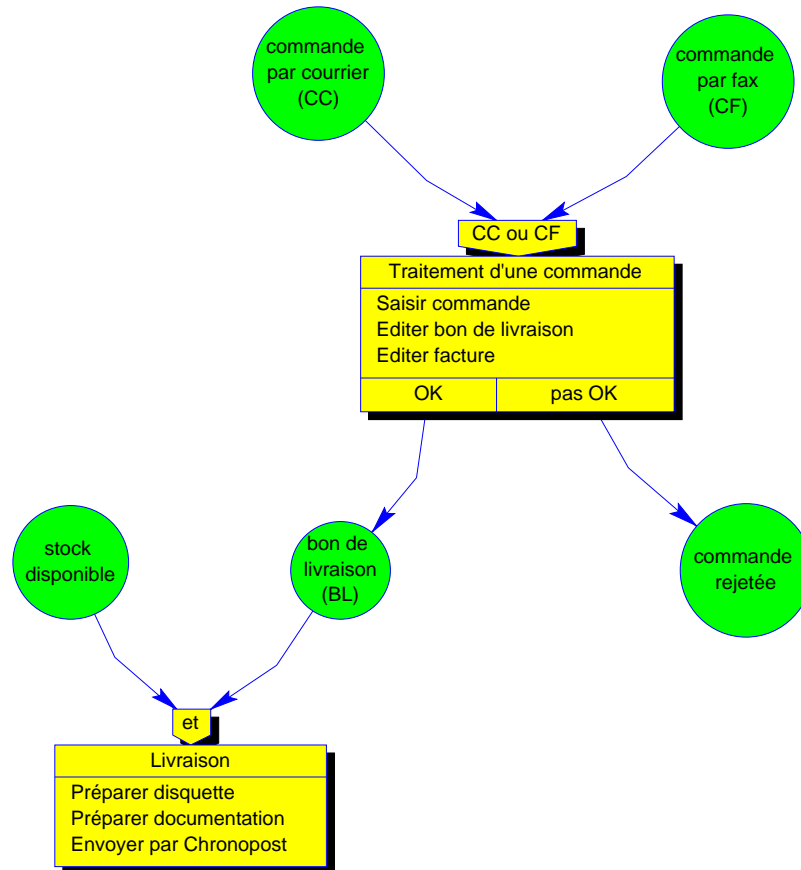
Exemples : Processus de facturation

Règles :

◇ Un domaine peut être représenté par un ensemble de processus.



Exemple de MCT :



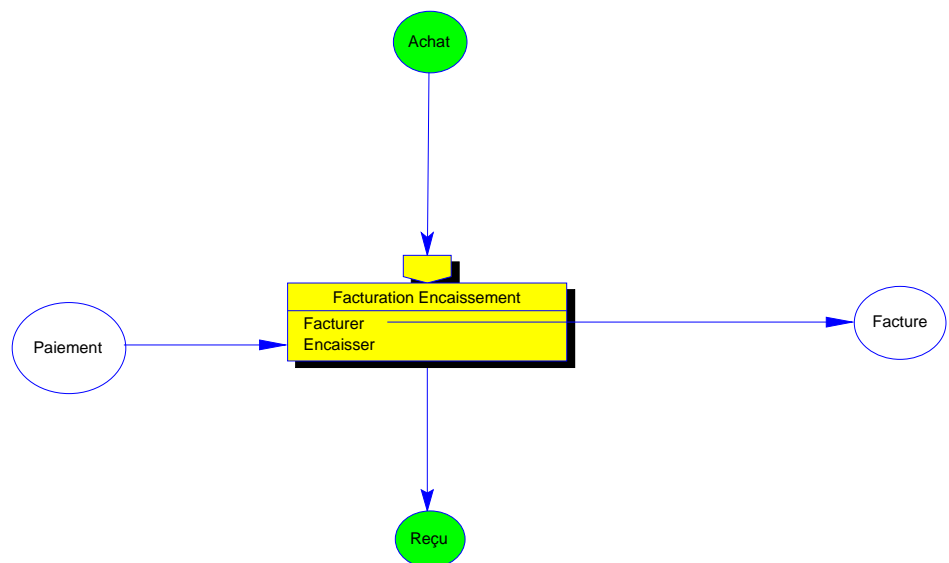
Quelques règles :

◆ Non interruption :

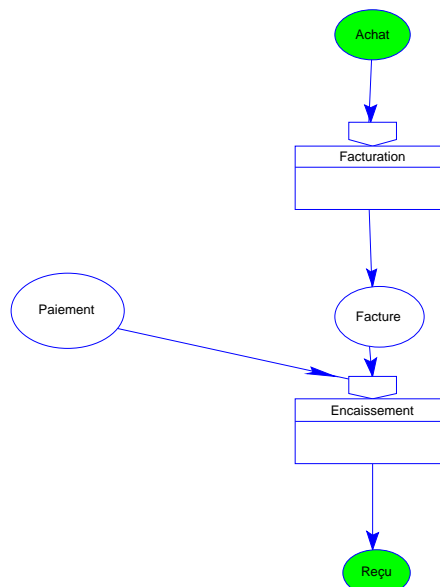
Une opération ne peut pas être interrompue par l'attente d'un événement externe.

→ Rajouter une deuxième opération déclenchée par cet événement externe.

Exemple : L'opération suivante :



Doit être décomposée en deux opérations :



◆ **Le temps :**

Le facteur temps ne doit être pris en compte dans le MCT que s'il ne constitue pas un élément organisationnel.

Exemple : Envoi d'une déclaration fiscale avant la fin d'exercice ; règle indépendante de l'organisation interne.

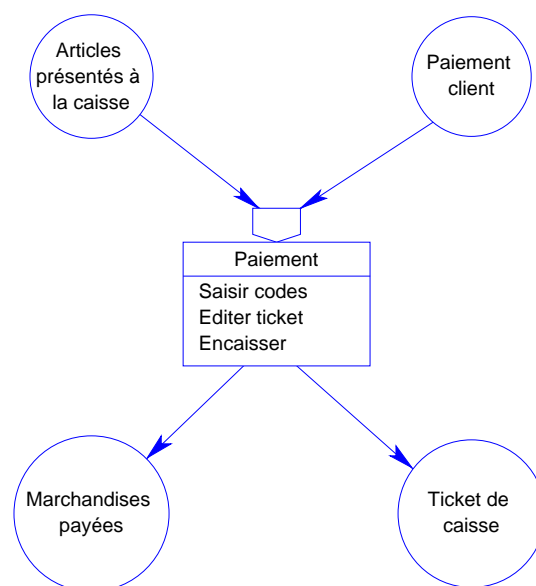
◆ **Événements intermédiaires significatifs :**

Lorsque certaines actions d'une opération ne peuvent être réalisées que lorsqu'un événement interne est réalisée, ce dernier est dit événement intermédiaire significatif (EIS).

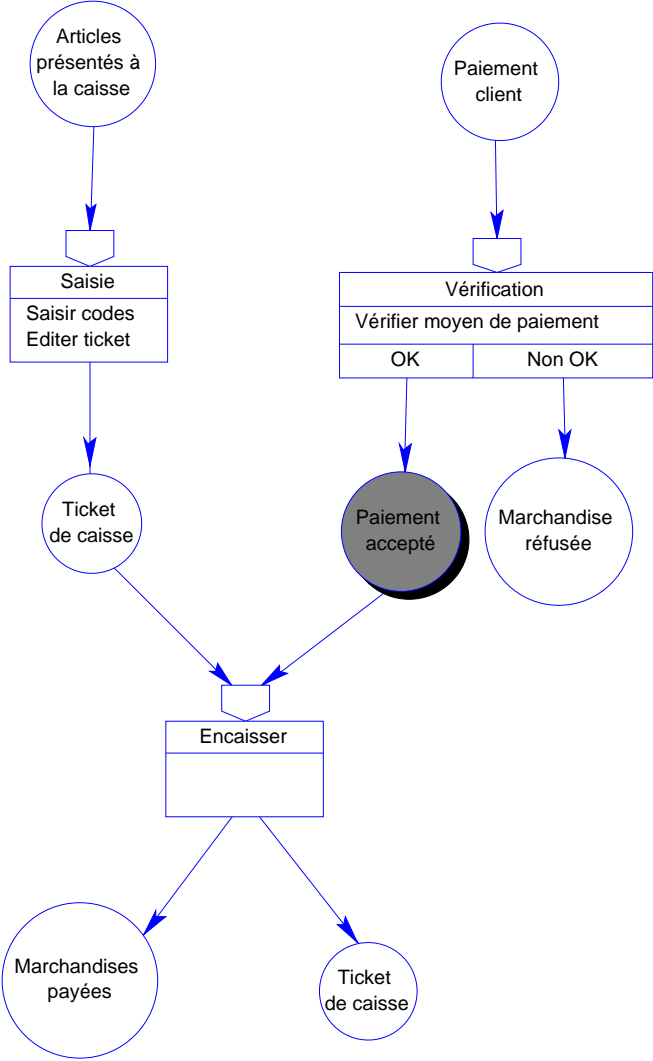
Un EIS doit être représenté dans le MCT.

→ décomposition de l'opération.

Exemple : L'opération suivante :



doit être découpée comme suit :



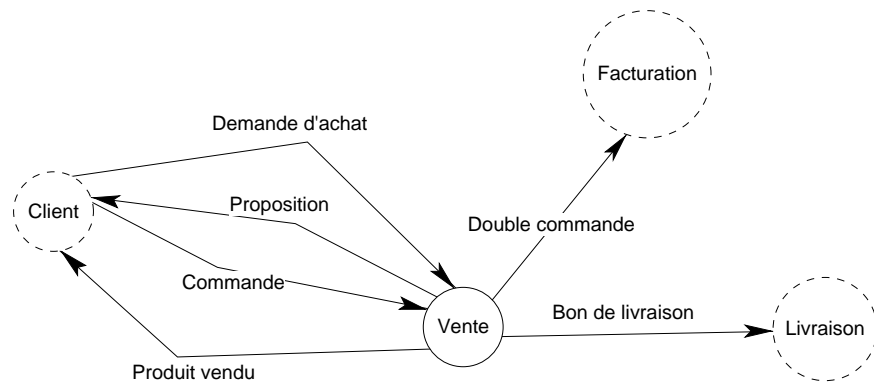
Démarche pour la construction d'un MCT :

La construction d'un MCT se fait selon les étapes suivantes :

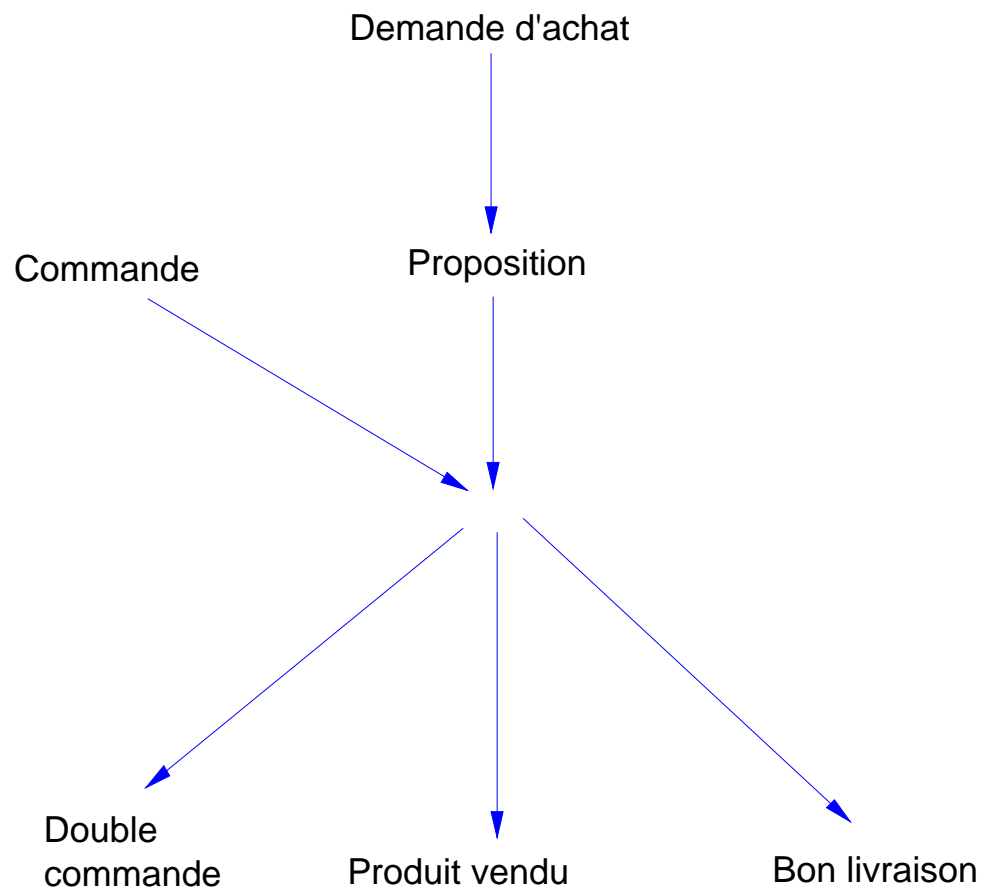
- 1) Construction d'un diagramme des flux entre les acteurs externes et internes pour chaque domaine.
- 2) Transformation des flux en graphe d'enchaînement des flux (événements/résultats).
- 3) Transformer ce graphe d'enchaînement des flux en MCT en remplaçant chaque passage d'un ensemble d'événements à un ensemble de résultats par une opération.

Exemple :

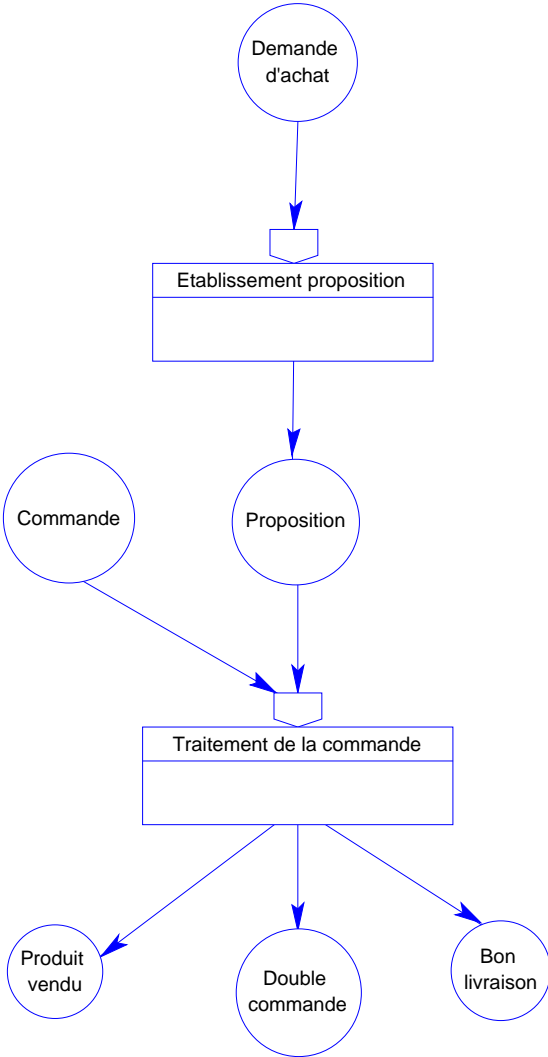
- *Diagramme des flux du domaine vente :*



- *Graphe d'enchaînement des événements résultats :*



• *MCT* :



3.5.2 Modèle organisationnel de traitements

Présentation :

- ◆ Permet d'effectuer une représentation qui tient compte des contraintes et règles **organisationnelles** de l'entreprise.
- ◆ Il permet de décrire le "QUAND", le "QUI" et le "OU".
- ◆ Les événements et les résultats du MCT sont maintenus.
- ◆ Les opérations sont remplacées par des **procédures fonctionnelles**.
- ◆ Les **postes de travail** effectuant les procédures fonctionnelles sont introduits.

Terminologie :

◆ Événement, résultat et synchronisation :

Mêmes règles que dans le MCT.

◆ Procédure :

Elle correspond à une opération ou à certaines actions d'une opération.

C'est la participation d'un poste de travail au traitement d'un ou plusieurs événements synchronisés.

Exemples : consultation stock

Règles :

- ◇ Une procédure peut être déclenchée par des événements internes ou externes.
- ◇ Une procédure est composée de tâches.
- ◇ Une procédure peut être manuelle, totalement automatisée ou partiellement automatisée.
- ◇ Une opération se décompose en une ou plusieurs procédures.
- ◇ Une procédure peut regrouper des traitements relatifs à plusieurs opérations.

- ◇ Plusieurs procédures peuvent contenir un même sous-ensemble de traitements (actions).
- ◇ Le déclenchement de la première procédure d'une opération suit les mêmes règles que le déclenchement de cette opération.

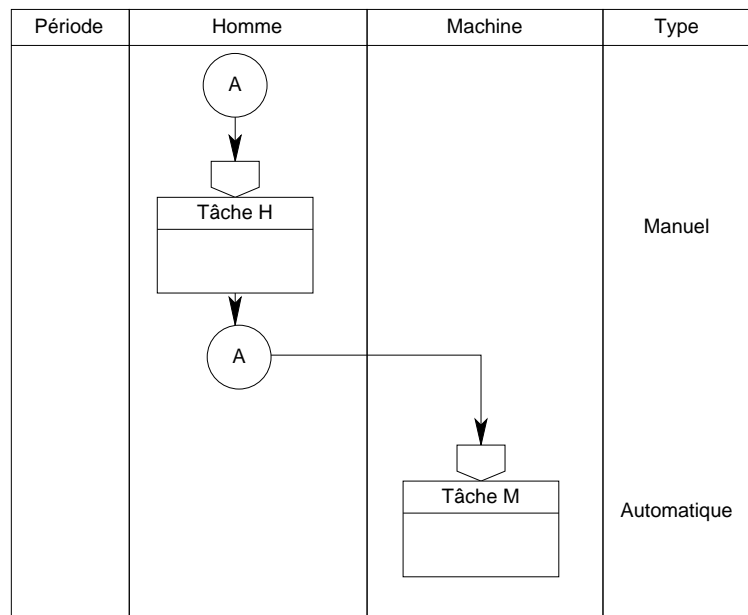
◆ **Tâche :**

C'est un ensemble de traitements effectués soit par la machine seule, soit alternativement par l'homme et la machine, dans le cadre d'une procédure.

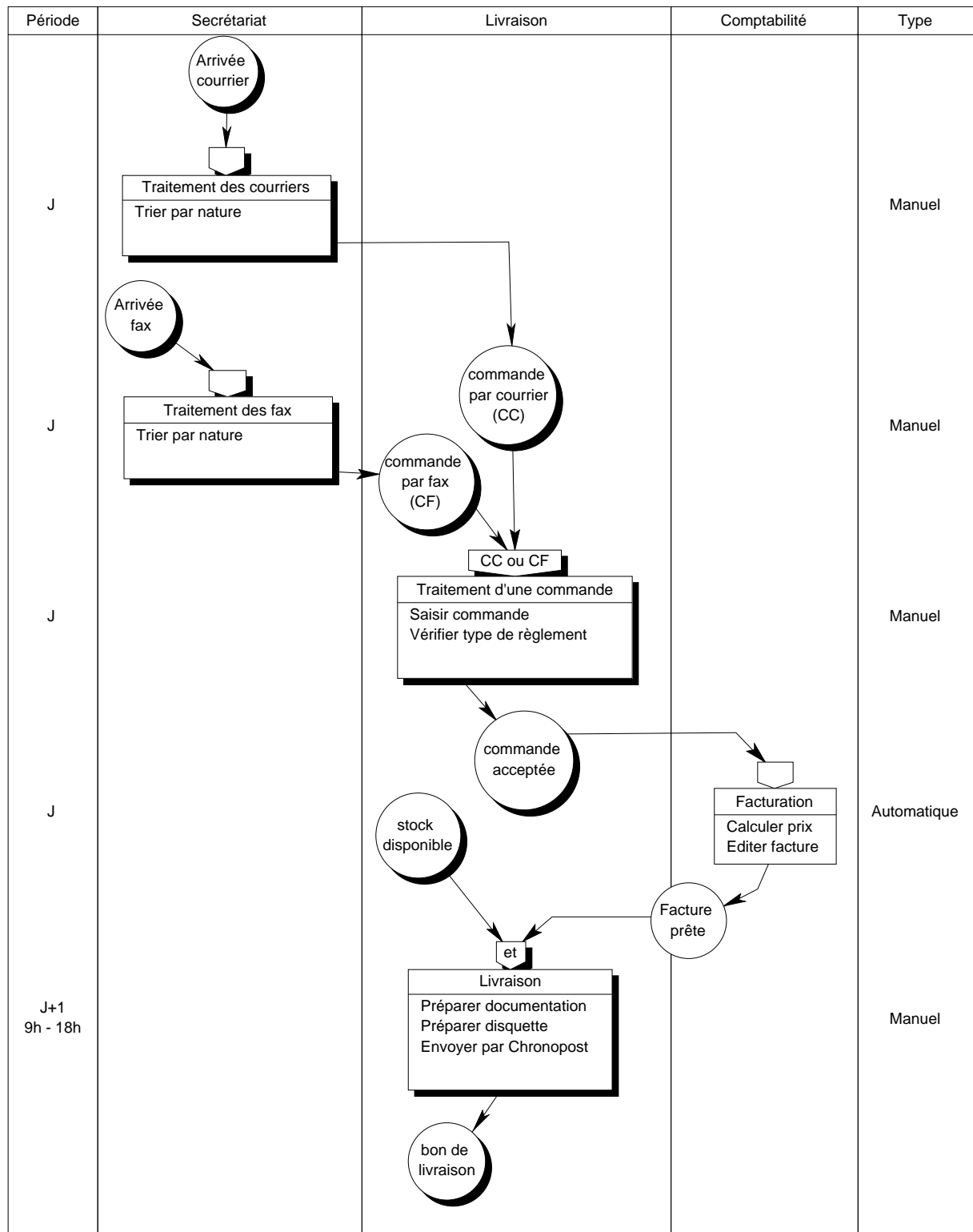
Exemples : Saisie du code article

Règles :

- ◇ Les tâches sont utilisées pour représenter, pour une procédure donnée, l'interaction homme/machine.

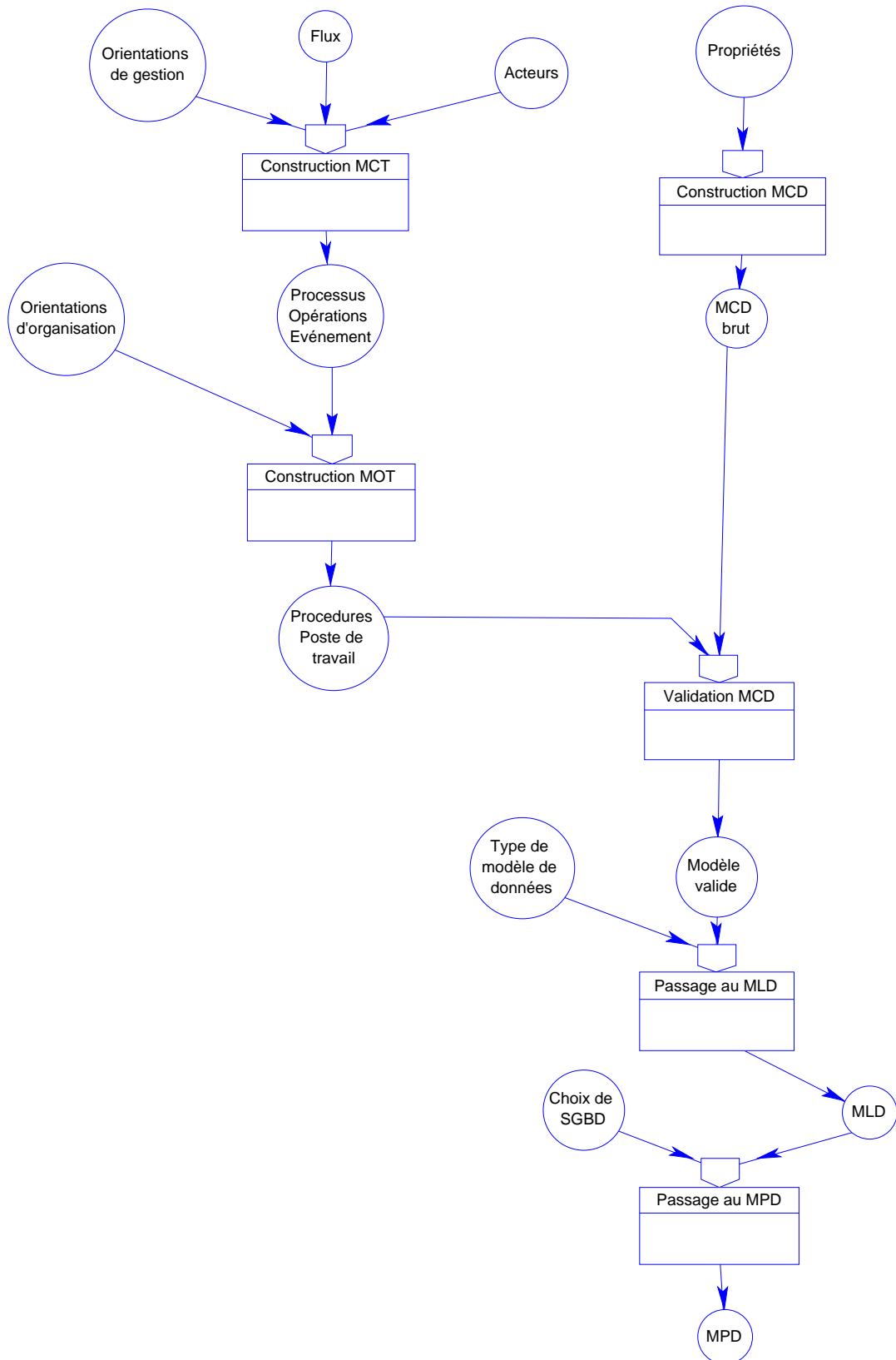


Exemple de MOT :



3.6 Résumé

Démarche pour la construction des modèles de base :



4. MERISE 2 : EXTENSION DE MERISE

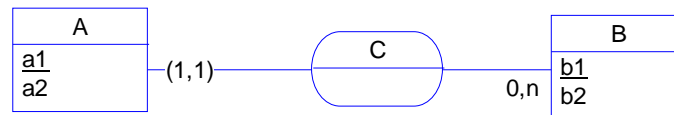
4.1 Introduction

- ◆ L'apparition des méthodes orientées objet et la complexité des systèmes d'information ont poussé vers l'extension de Merise pour rallonger sa durée de vie :
 - ◇ OOM
 - ◇ Merise 2
- ◆ **Principales extensions de Merise 2:**
 - ◇ Notion de *lien identifiant*.
 - ◇ Notion d'*association d'associations*.
 - ◇ Introduction de l'*héritage* au niveau des données.
- ◆ Pour chacune de ces extensions, nous allons donner :
 - ◇ Le principe au niveau conceptuel.
 - ◇ Un exemple.
 - ◇ La conséquence sur le modèle logique.

4.2 Notion de lien identifiant

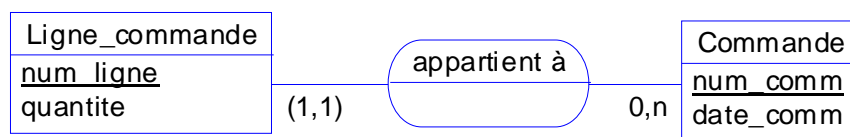
4.2.1 Principe au niveau conceptuel

- ◆ C'est une association binaire ayant les cardinalités « (1,1) » et « x, n », x pouvant être 0 ou 1.

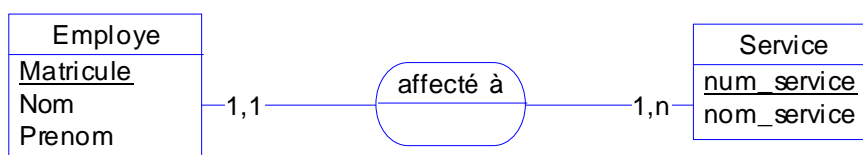


- ◆ L'entité *A* est dite « **entité faible** » et l'entité *B* « **entité forte** »
- ◆ Chaque occurrence de *A* ne peut exister que s'il existe une occurrence de *B* qui lui est associée.
- ◆ La suppression d'une occurrence de *B* entraîne la suppression de toutes les occurrences de *A* qui lui sont associées (suppression en cascade).
- ◆ L'identifiant *a1* de *A* est relatif : il ne différencie que les occurrences de *A* correspondant à une occurrence de *B*.

4.2.2 Exemples



- ◆ Contre-exemple :

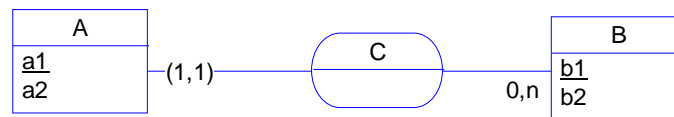


4.2.3 Traduction d'un lien identifiant au niveau MLD

◆ Une association de type lien identifiant se traduit de la façon suivante :

1. Application de la règle relative aux associations binaires ayant les cardinalités « x, 1 » et « x, n »
2. L'identifiant qui migre de l'entité ayant la cardinalité la plus élevée vers celle ayant la cardinalité la plus faible fait partie de la clé primaire.

◆ L'association suivante :



se traduit comme suit :

A (b1#, a1, a2)

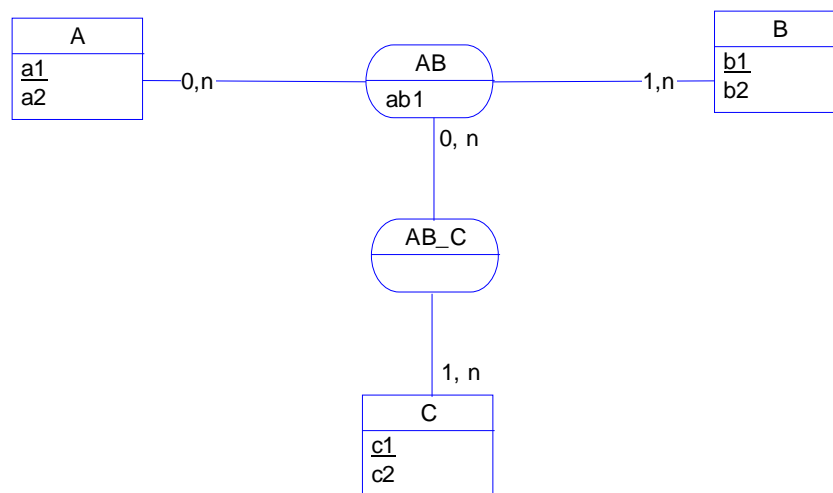
B (b1, b2)

◆ Au niveau physique, la clé étrangère *b1* doit être définie avec l'option « *on delete cascade* ».

4.3 Notion d'association d'associations

4.3.1 Principe au niveau conceptuel

- ◆ Dans Merise de base, seules les entités peuvent participer à une association.
- ◆ Dans Merise 2, une association peut participer à une autre association.

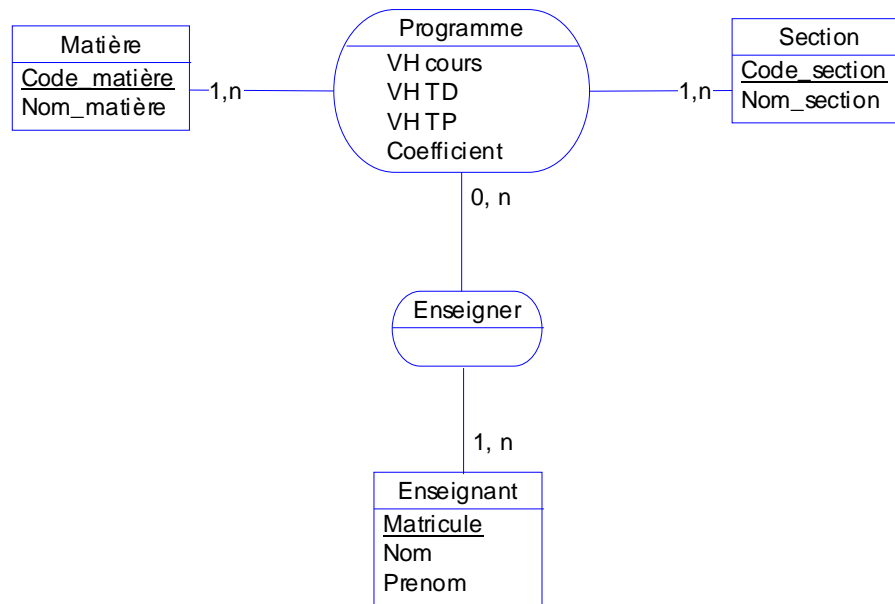


- ◆ L'association AB relie A et B : c'est une association ordinaire
- ◆ L'association AB_C relie l'entité C à l'association AB : c'est une association d'association.

Remarque :

Pour qu'une association puisse participer à une autre association, il faut qu'elle soit une association de dimension 3 ou plus ou bien une association binaire ayant les cardinalités « x, n » et « x, n »

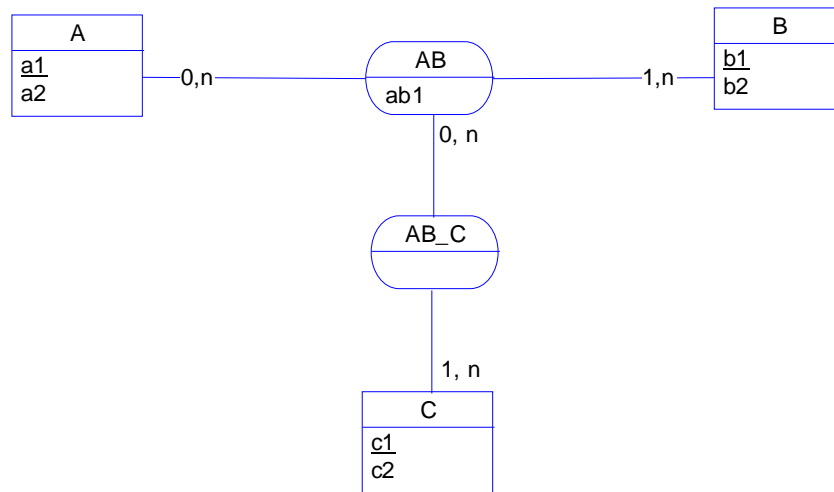
4.3.2 Exemple



4.3.3 Traduction d'une association d'associations au niveau MLD

- ◆ La traduction d'une association d'associations au niveau MLD se fait selon les étapes suivantes :
 1. Traduction de l'association qui participe à l'association. Celle-ci donne lieu à une table car elle a est de dimension 3 ou plus ou bien binaire et ayant les cardinalités « x, n », « x, n ».
 2. Traduction de l'association d'association en considérant l'association participante comme une entité et en appliquant les règles de passage d'un MCD vers un MLD.

◆ L'association suivante :



se traduit comme suit :

A (a1, a2)

B (b1, b2)

C (c1, c2)

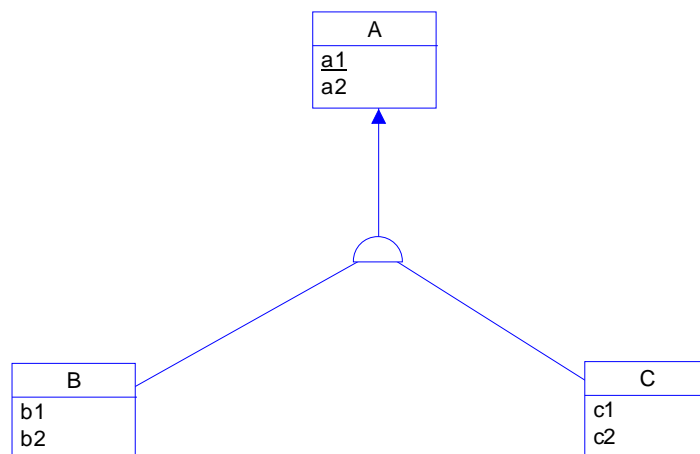
AB (a1#, b1#, ab1)

AB_C ([a1, b1]#, c1#)

4.4 Notion d'héritage

4.4.1 Principe au niveau conceptuel

- ◆ Merise 2 a intégré la notion d'héritage au niveau de données.
- ◆ L'héritage est représenté comme suit :



- ◆ A est dite « super entité » et B et C « sous entités ».
- ◆ Cette association signifie que les entités B et C héritent les propriétés et les associations de l'entité A.
- ◆ L'association d'héritage peut être :
 - ❖ **Exclusive** (\cup^x) : une occurrence de A ne peut avoir qu'une seule occurrence au niveau de B ou C.
 - ❖ **Non exclusive** (\cup) : une occurrence de A peut avoir une occurrence au niveau de B et une autre au niveau de C.

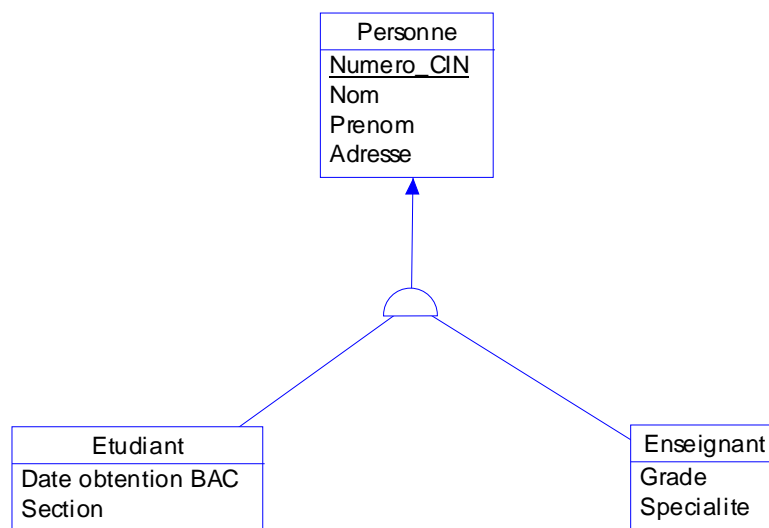
◆ Une association d'héritage peut être obtenue :

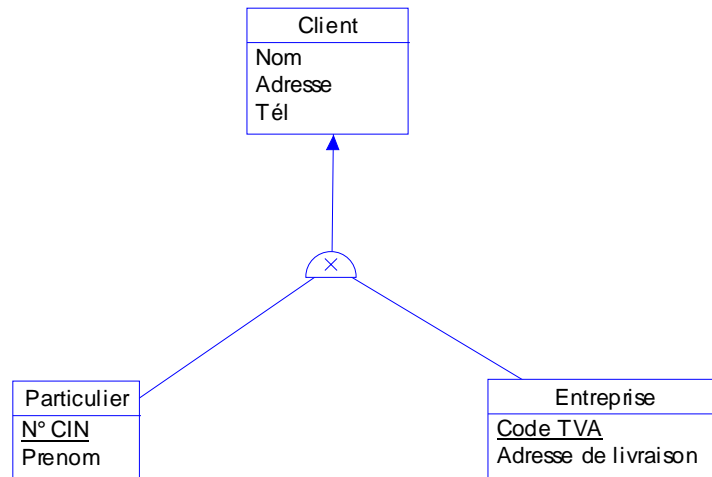
- ❖ **Par spécialisation** : Une entité est éclatée en deux ou plusieurs autres entités. Les propriétés communes restent au niveau de la super entité et chaque sous entité contient ses propriétés spécifiques
- ❖ **Par généralisation** : Si deux ou plusieurs entités comportent des propriétés communes et/ou des associations communes, celles-ci seront regroupées dans une nouvelle entité qui sera la super entité.

Remarque :

Dans une association d'héritage, on ne peut pas avoir des identifiants à la fois au niveau de la super entité et des sous entités. Si l'identifiant fait partie des propriétés communes, il doit se situer au niveau de la super entité, sinon, chaque sous entité doit avoir son propre identifiant.

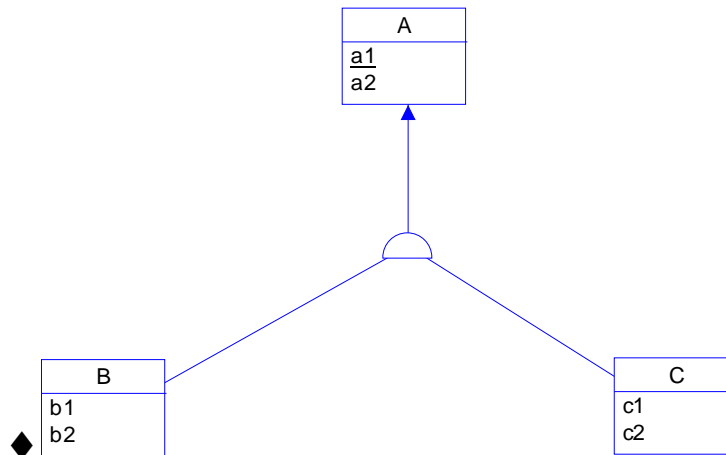
4.4.2 Exemple





4.4.3 Traduction de l'héritage au niveau MLD

◆ L'association d'héritage suivante



peut être faite de trois façons :

1. Traduction dite « Mère uniquement » ou « Généralisation » :

A (a1, a2, b1, b2, c1, c2)

2. Traduction dite « Filles uniquement » ou « Spécialisation » :

B (a1, a2, b1, b2)

C (a1, a2, c1, c2)

3. Traduction dite « Mère et filles » :

A (a1, a2)

B (a1, b1, b2)

C (a1, c1, c2)

4.5 Modèle conceptuel de traitements analytique

4.5.1 Principe au niveau conceptuel

- ◆ Permet de formaliser les liens entre les données (MCD) et les traitements (MCT).
- ◆ Formalisme issu de la fusion de ceux du MCD et du MCT

5. MÉTHODES DE CONCEPTION ORIENTÉES OBJET

5.1 Introduction

Caractéristiques des méthodes de conception orientées objet :

- ◆ Elles constituent une évolution des méthodes systémiques vers une **plus grande cohérence entre les objets et leur dynamique.**
- ◆ Elles sont basées sur le concept d'**objet.**
- ◆ Elles permettent de décrire la **dynamique** du SI comme un ensemble d'**opérations rattachées aux objets** constituant le système.
- ◆ Cette représentation permet une meilleure modularité et une réutilisation des composants du SI.
- ◆ C'est une approche ascendante :
 - ◇ Identification des objets de base du SI.
 - ◇ Par composition, constitution d'objets de plus en plus complexes.

Quelques méthodes orientées objet :

Méthodes	Auteurs
OOD	G. Booch
HOOD	Hood Technical Group
OOA	S. Shlear et S. Mellor
OOA / OOD	T. Coal et E. Yourdon
OMT	J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen
OOSE	I. Jacobson, M. Cristerson, P. Jonson, G. Övergaard
OOM	M. Bouzeghoub et A. Rochfeld
UML	OMG (Object Management Group)

5.2 Généralités

5.2.1 Concepts de base


◆ Notion d'objet :

Un **objet** est la représentation d'un concept abstrait ou une abstraction d'un objet physique du monde réel.

Un objet porte :

- ◇ des **attributs** représentant ses propriétés statiques.
- ◇ des **méthodes** représentant son comportement.
- ◇ une **identité** permettant de le distinguer des autres objets.

Exemple :

	Attributs : <i>N° immatriculation : 1254 XX 99</i> <i>Marque : Renault</i> <i>Type : Laguna</i>
	Méthodes : <i>Démarrer</i> <i>Accélérer</i> <i>Freiner</i>
	Attributs : <i>"La voiture de M. Untel"</i>

◆ **Notion de classe :**

Une **classe** contient la description des attributs et des méthodes qui caractérisent les objets rattachés à cette classe. On peut l'assimiler à un moule servant à fabriquer des objets.

Les objets rattachés à une classe sont les **instances** de cette classe.

Exemples : Employé, voiture, commande, etc.

◆ **Encapsulation :**

Permet de masquer aux utilisateurs d'un objet tous les détails relevant de son implémentation (partie **privée**) et de ne laisser accessible que la vue externe (**interface**).

L'encapsulation:

- ◇ garantie la sécurité et l'intégrité des données
- ◇ augmente la maintenabilité en limitant la portée des modifications

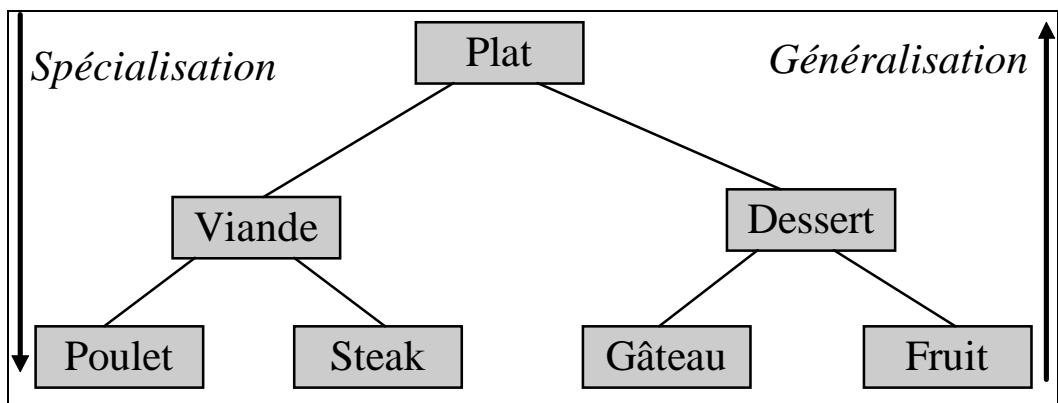
Exemple : Circuit intégré

◆ Généralisation / Spécialisation :

La **généralisation** consiste à regrouper au sein d'une **superclasse** les caractéristiques communes à un ensemble de classes (attributs et méthodes).

La **spécialisation** d'une **sous-classe** consiste à adapter les caractéristiques transmises par la superclasse et à lui ajouter des nouvelles caractéristiques.

La généralisation et la spécialisation génèrent une **hiérarchie** des classes.



◆ Héritage :

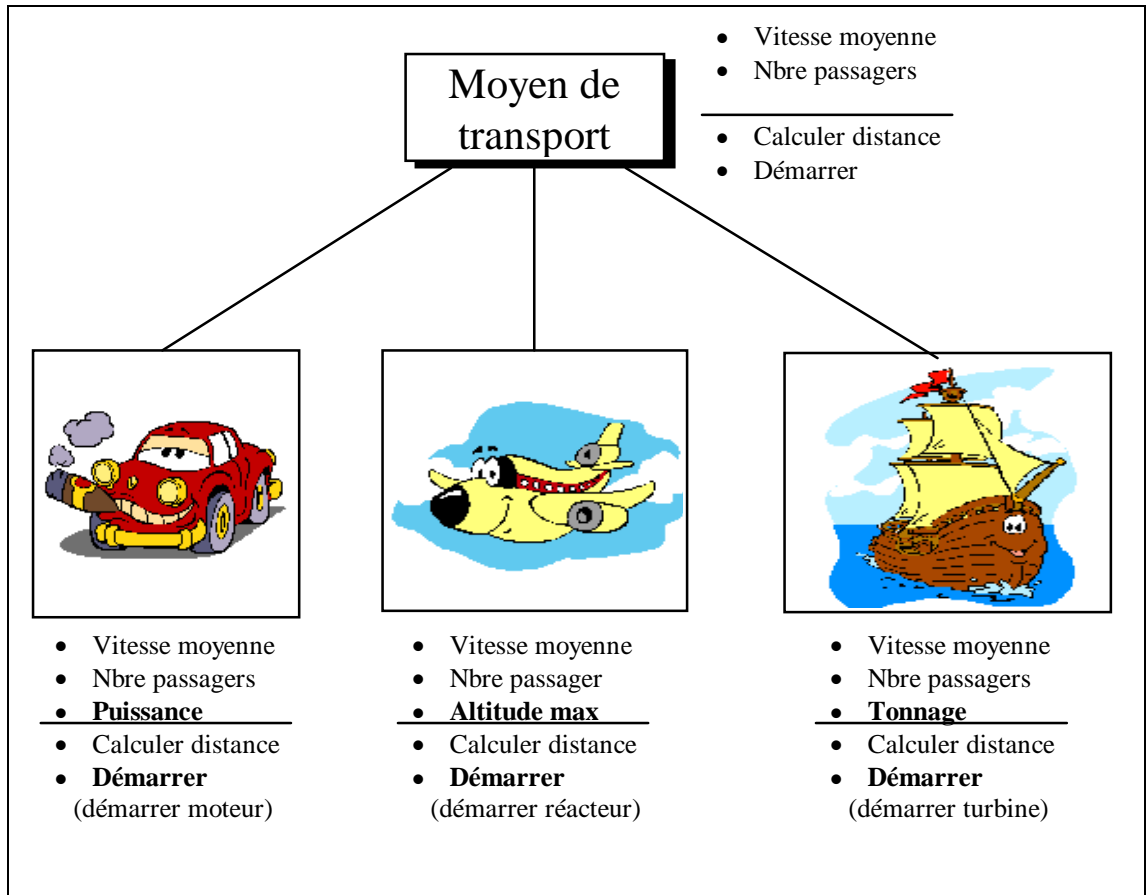
L'**héritage** permet le transfert des caractéristiques d'une superclasse vers ses sous-classes.

Une classe **hérite** des attributs et des méthodes de tous ses ancêtres.

L'héritage constitue un premier moyen de **réutilisation**.

◆ Polymorphisme :

C'est la capacité des objets d'une même hiérarchie de classes de répondre différemment à la même opération.



◆ Modularité :

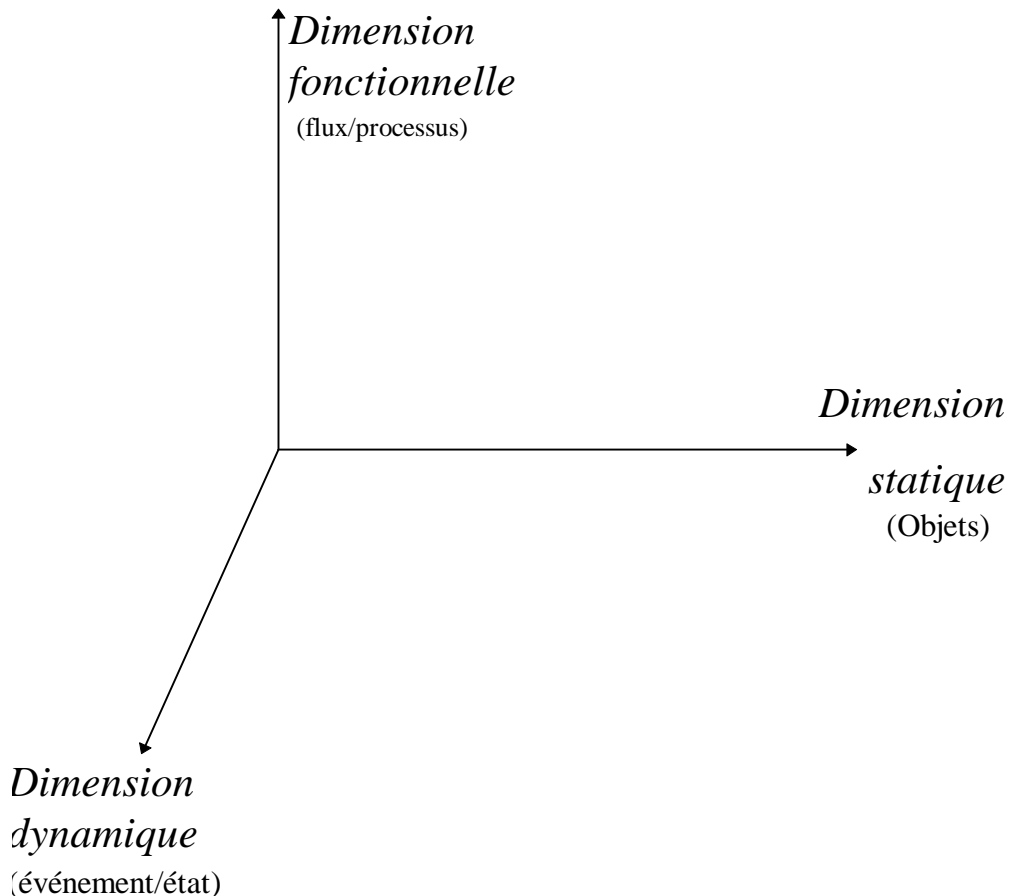
La **modularité** d'un système est réalisée si toutes les données et méthodes relatives à ces données sont **regroupées** au sein d'une même unité :

◇ une **classe** est un ensemble modulaire

◇ on peut aussi constituer un module en associant plusieurs classes (on parle alors de **sous-système**)

5.2.2 Les trois dimensions du SI

La plupart des méthodes objet ont une approche commune basée sur une triple perception du système d'information :



- ◆ **dimension statique** : décrit les **objets** du système, les **associations** entre ces objets, les **contraintes** et les **opérations** correspondantes.
- ◆ **dimension dynamique** : représente les types **d'événements** qui peuvent survenir dans le SI et les **changements d'états** résultant du traitement de ces événements.

◆ **dimensions fonctionnelles** : représente les flux d'informations qui circulent entre les différents acteurs du SI, ainsi que les processus qui les transforment.

Position des différentes méthodes par rapport aux trois dimensions :

Dimension	OOD	HOOD	OOA/ OOD	OMT	OOSE	OOM
Statique	✓	✓	✓	✓	✓	✓
Dynamique	✓			✓	✓	✓
Fonctionnelle				✓		✓

5.3 Méthode OMT

5.3.1 Présentation générale

- ◆ OMT : **Object Modeling Technique**
- ◆ Développée chez Général Electric au Research and Development Center (GE R&D).
- ◆ Auteurs de la méthode
 - ◇ J. Rumbaugh (spécialiste langages OO)
 - ◇ M. Blaha (spécialiste conception de bases de données)
 - ◇ W. Premerlani (spécialiste SGBD)
 - ◇ F. Eddy (spécialiste architecture des applications temps réel)
 - ◇ W. Lorensen (spécialiste intelligence artificielle et simulation)
- ◆ Publication en 1990 du livre référence de la méthode:
Object Modeling and design

5.3.2 Principes de la méthode

- ◆ OMT permet la modélisation de systèmes complexes selon 3 points de vue :
 - ◇ Aspect **statique** : → Modèle **objet**
 - ◇ Aspect **dynamique** : → Modèle **dynamique**
 - ◇ Aspect **fonctionnel** : → Modèle **fonctionnel**

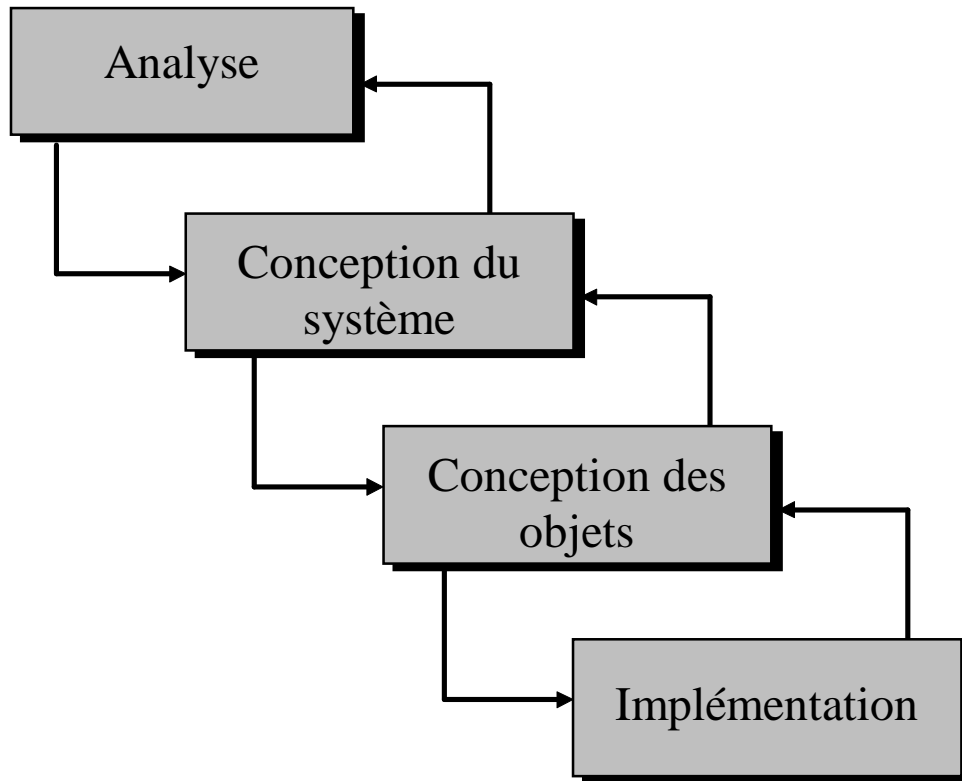
- ◆ Il existe des règles de cohérence entre les trois modèles.

- ◆ Chacun des trois modèles est décrit grâce à un formalisme qui lui est propre. Ce formalisme est conservé pour chaque phase des cycles d'analyse/conception.

- ◆ Les différents modèles sont enrichis au fur et à mesure de l'avancement du projet.

5.3.3 Démarche méthodologique

OMT utilise le modèle en cascade :



Phase d'analyse :

Permet d'élaborer les trois modèles conceptuels : *statique, dynamique et fonctionnel*.

Permet de décrire **ce que** le système doit faire et non pas **comment** le faire.

Elle est composée des étapes suivantes :

◆ Rédiger ou obtenir une **description initiale du problème**

◆ Construire un **modèle objet** :

- ◇ Identifier les classes
- ◇ Constituer un dictionnaire de données (description des classes, attributs et association)
- ◇ Ajouter les associations entre classes
- ◇ Ajouter les attributs aux classes et aux liens
- ◇ Réorganiser et simplifier les classes en utilisant l'héritage
- ◇ Tester les chemins d'accès en utilisant des scénarios (itérations)
- ◇ Regrouper les classes en modules en se basant sur les fonctions inter reliées.



Diagramme de modèle objet

+

dictionnaire de données

◆ Construire un **modèle dynamique** :

- ◇ Préparer des séquences typiques d'interaction
- ◇ Identifier les événements entre les objets et préparer une base d'événements pour chaque scénario
- ◇ Préparer un diagramme de flux d'événements pour le système
- ◇ Développer un diagramme d'états pour chaque classe ayant un comportement dynamique important
- ◇ Vérifier la consistance et la complétude des événements partagés dans les diagrammes d'état.



Diagramme d'état

+

diagramme global de flux d'événements

◆ Construire un **modèle fonctionnel** :

- ◇ Identifier les valeurs en entrée et en sortie
- ◇ Utiliser les diagrammes de flux de données si nécessaire pour montrer les dépendances fonctionnelles
- ◇ Décrire le rôle de chaque fonction
- ◇ Identifier les contraintes
- ◇ Spécifier les critères d'optimisation



Diagramme de flux de données (DFD)

+

contraintes

◆ Vérifier, réitérer et raffiner les trois modèles :

- ◇ Rajouter dans le modèle objet les principales opérations découvertes lors de l'élaboration du modèle fonctionnel (ne pas surcharger le MO par toutes les opérations).
- ◇ Vérifier que les classes, associations et opérations sont consistantes et complètes à ce niveau d'abstraction (tester les modèles à travers des scénarios)
- ◇ Développer des scénarios plus détaillés (avec des erreurs)
- ◇ Réitérer toutes ces étapes jusqu'à la fin de l'analyse.



Documents d'analyse validés :

Position du problème

+

Modèle objet

+

Modèle dynamique

+

Modèle fonctionnel

Phase de conception du système :

Il s'agit ici d'apporter les premiers éléments pour résoudre les problèmes posés en phase d'analyse.

Elle est composée des étapes suivantes :

◆ Estimation des performances et des ressources nécessaires :

- ◇ Exécuter des calculs simples et des expérimentations pour estimer les besoins en taille et en performances.
- ◇ Utiliser ces estimations pour déterminer les points critiques et fixer les choix de conception.
- ◇ Réaliser une maquette du système (ou d'une partie du système) si nécessaire.

◆ Décomposition du système :

- ◇ Définir des sous-systèmes en fonction de points communs (fonctionnalités, localisation, hardware, performances, etc.)
- ◇ Les sous systèmes doivent regrouper des classes, des associations, des événements et des contraintes qui sont en relation.
- ◇ L'architecture du système peut être organisée en couches horizontales et/ou en partitions verticales.

◆ Gestion de la concurrence :

- ◇ Identifier les objets qui agissent simultanément : les **agents**.
- ◇ Identifier le rôle de chaque agent : **acteur** ou **recevant**

◆ Allocation et gestion des ressources :

- ◇ Définir la stratégie de stockage des données (BD, fichiers).
- ◇ Estimation des performances requises et identifier les ressources nécessaires pour les satisfaire.
- ◇ Allocation des ressources aux sous-systèmes.

◆ Choix des priorités d'optimisation et d'échange :

- ◇ Déterminer l'importance relative des critères d'optimisation

Phase de conception objet :

Il s'agit de définir l'implémentation des classes et des associations, ainsi que les interfaces internes et algorithmes des méthodes utilisées pour implémenter les opérations.

Elle consiste à :

- ◆ Déplacer les opérations des modèles fonctionnel et dynamique vers le modèle objet :

◇ Il s'agit d'identifier les opérations dans le modèle fonctionnel (processus) et dynamique (événements, actions, activités) et les intégrer dans le modèle objet.

- ◆ Identifier et définir les sous-opérations

◇ Une sous opération est une partie d'une opération qui ne peut pas être appelée de l'extérieur de la classe

◇ Elles permettent d'identifier les parties réutilisables.

◇ Elles seront privées

◆ Identifier et définir les classes de conception

◇ Il s'agit des classes d'interface (entre sous-systèmes et entre le système et les utilisateurs), des classes de contrôle du comportement des autres classes et des méta classes (décrivant la structure d'autres classes)

◆ Concevoir et affiner les algorithmes et les structures de données

◇ Il s'agit de choisir des algorithmes et des structures de données efficaces pour mettre en œuvre les opérations

◆ Optimiser le modèle objet

◇ Il s'agit de restructurer le modèle objet pour rendre son implémentation simple et efficace

◆ Affiner la structure de classe avec l'héritage :

◇ Rechercher des structures similaires afin d'accroître la réutilisation du code

◇ Casser éventuellement des liens sémantiques entre classes.

◆ Décomposer la conception en modules physiques

◇ Il s'agit d'identifier des modules physiques à partir du découpage effectué.

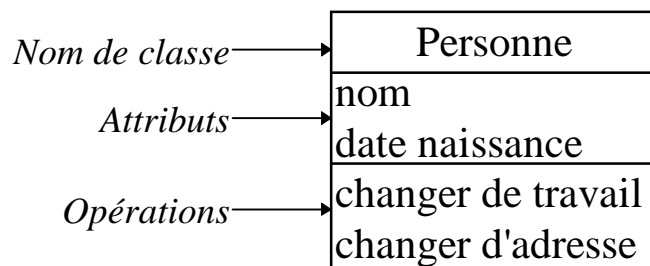
5.3.4 Modélisation

5.3.4.1 Modèle objet

Permet d'effectuer une représentation **statique** d'un système en décrivant les classes d'objets, les relations entre classes et les attributs et les opérations qui caractérisent chaque classe.

Classe :

Représentation initiale (simple) :



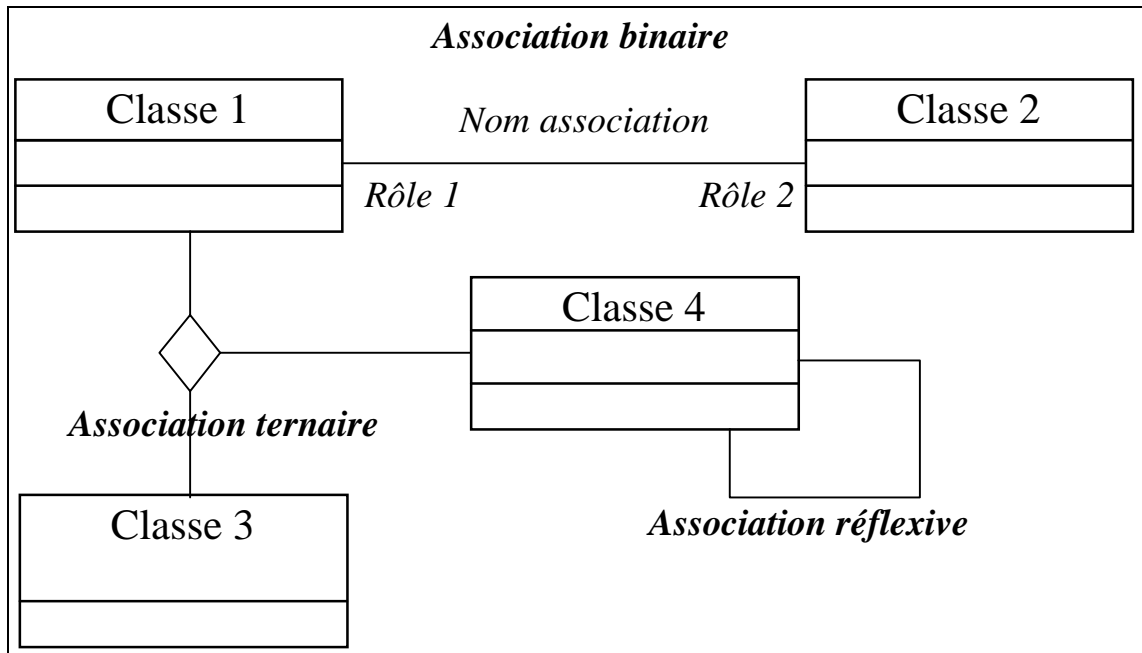
Représentation complète :

Nom classe
attribut1 : type de données = valeur par défaut
attribut2 : type de données = valeur par défaut
opération1(liste arguments) : type résultat
opération2(liste arguments) : type résultat

Remarque : Les identificateurs internes des objets ne doivent pas être listés avec les attributs, par contre les attributs ayant une existence réelle doivent apparaître (matricule, N° sécurité sociale, etc.).

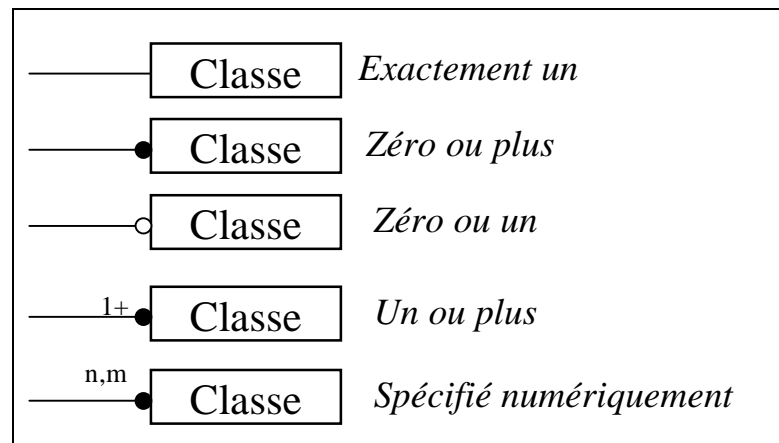
Associations :

Forme générale :

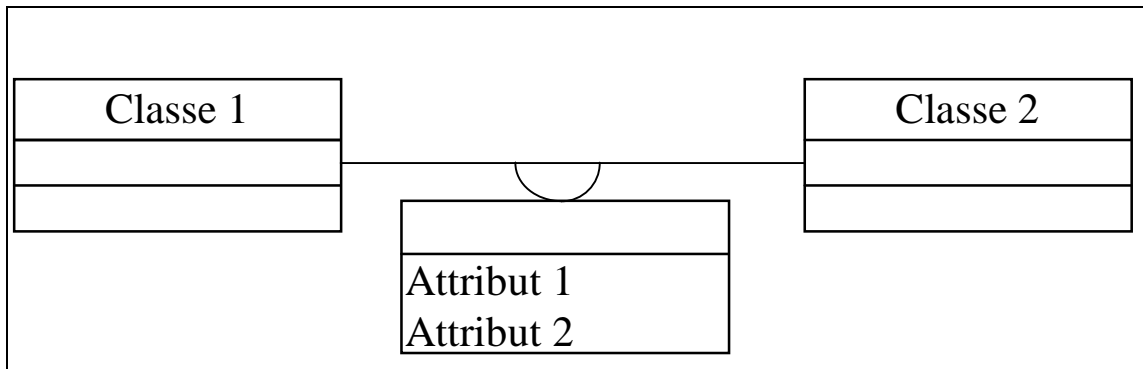


Cardinalités :

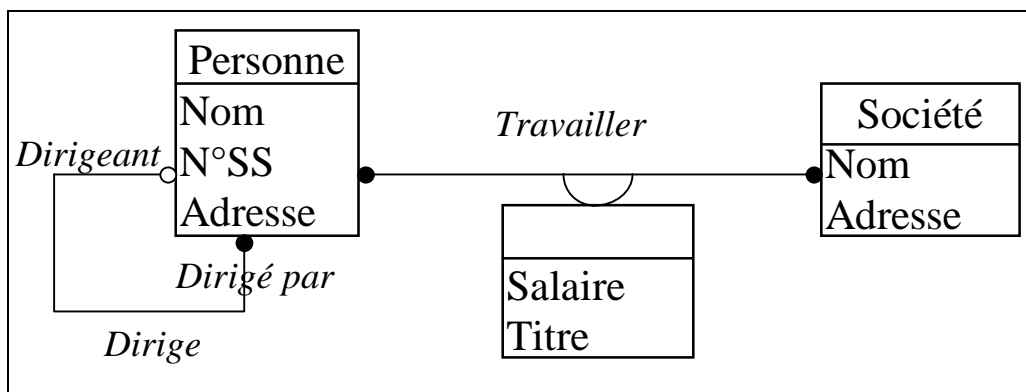
Les cardinalités sont représentées par des symboles graphiques ou spécifiées numériquement.



Association porteuse de données :

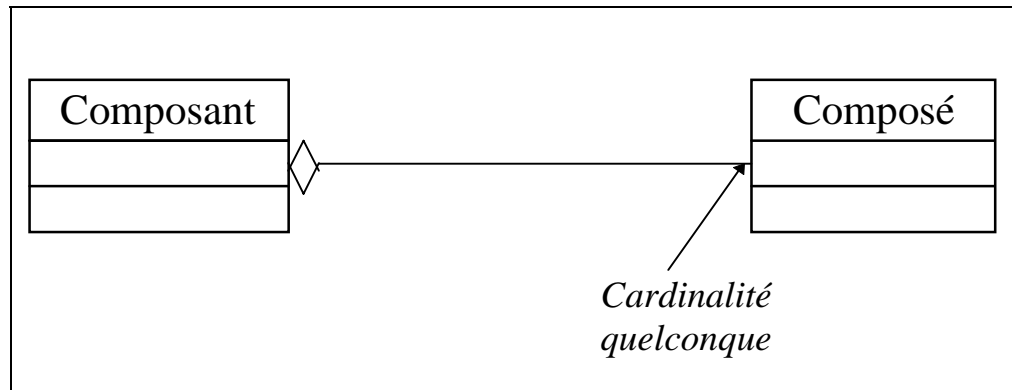


Exemple :

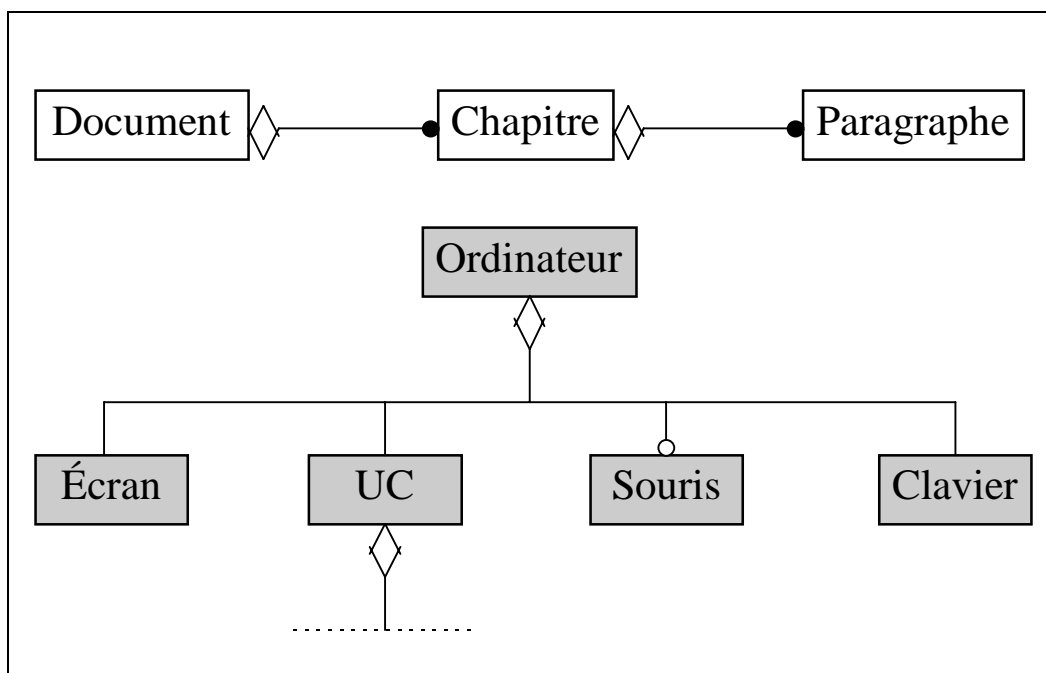


Agrégation :

C'est une forme particulière d'association permettant d'exprimer une relation "**Composant-composé**".

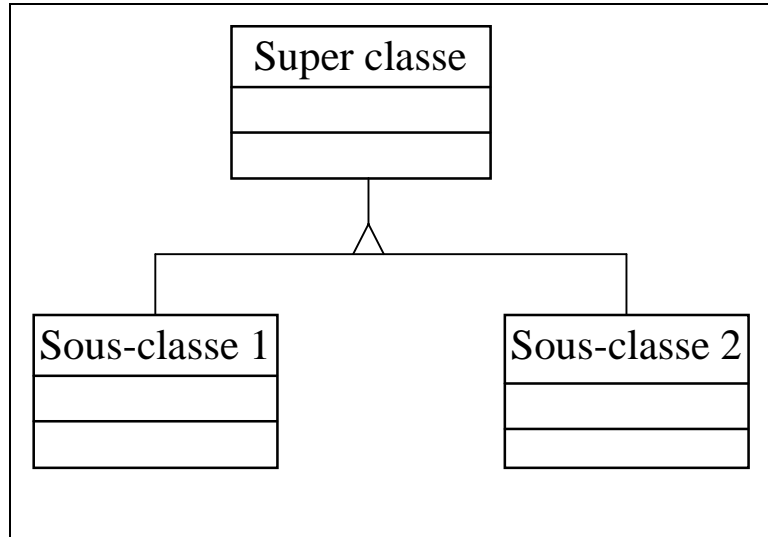


Exemples :

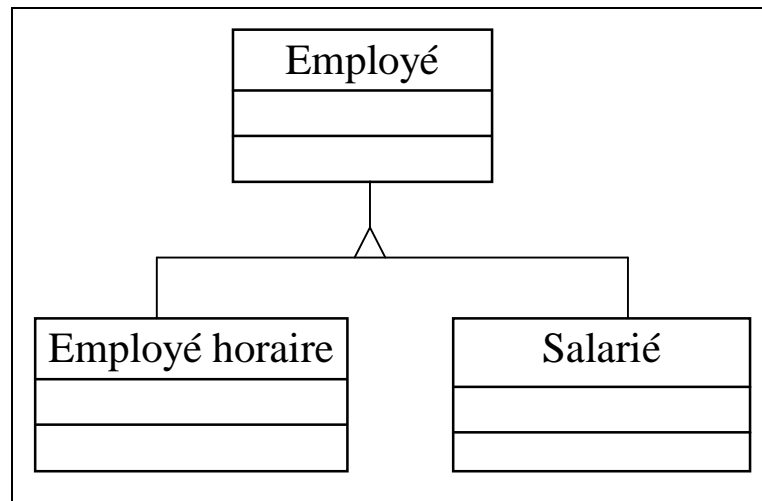


Héritage :

Permet de représenter le partage par deux ou plusieurs classes d'un certain nombre de caractéristiques communes.



Exemple :



5.3.4.2 Modèle dynamique

Permet de décrire le cycle de vie des objets à travers les changements d'états possibles.

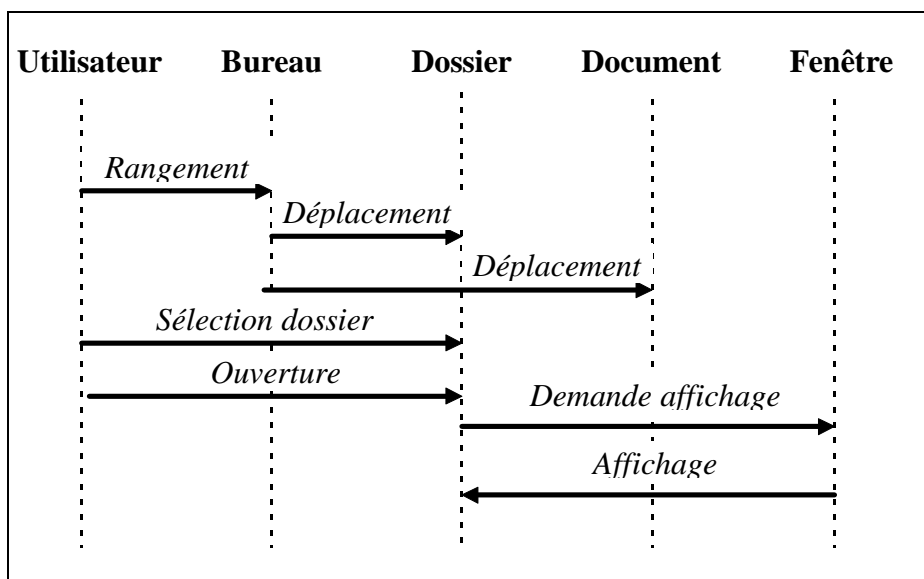
Le modèle dynamique repose sur deux types de diagrammes qui sont :

◇ un **diagramme de trace d'événements** (event trace) pour chaque scénario d'utilisation du système ou **cas d'utilisation** (use case).

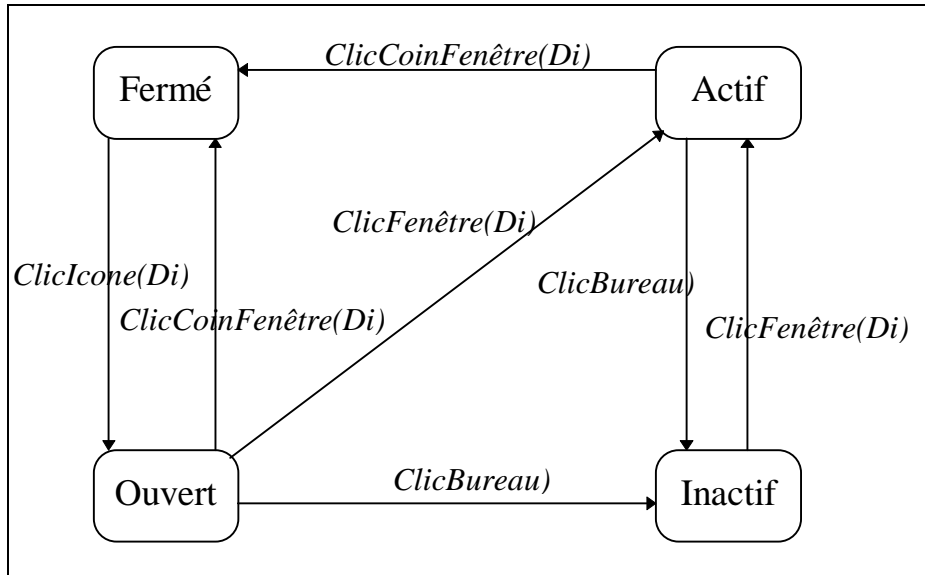
Il n'y a pas d'alternatives dans un scénario → un autre scénario.

◇ un **diagramme d'état** pour chaque classe montrant les **événements** que reçoit et émet la classe et les différents **états** par lesquels elle passe.

Exemple de diagramme de trace d'événements :



Exemple de diagramme d'états pour la classe document :

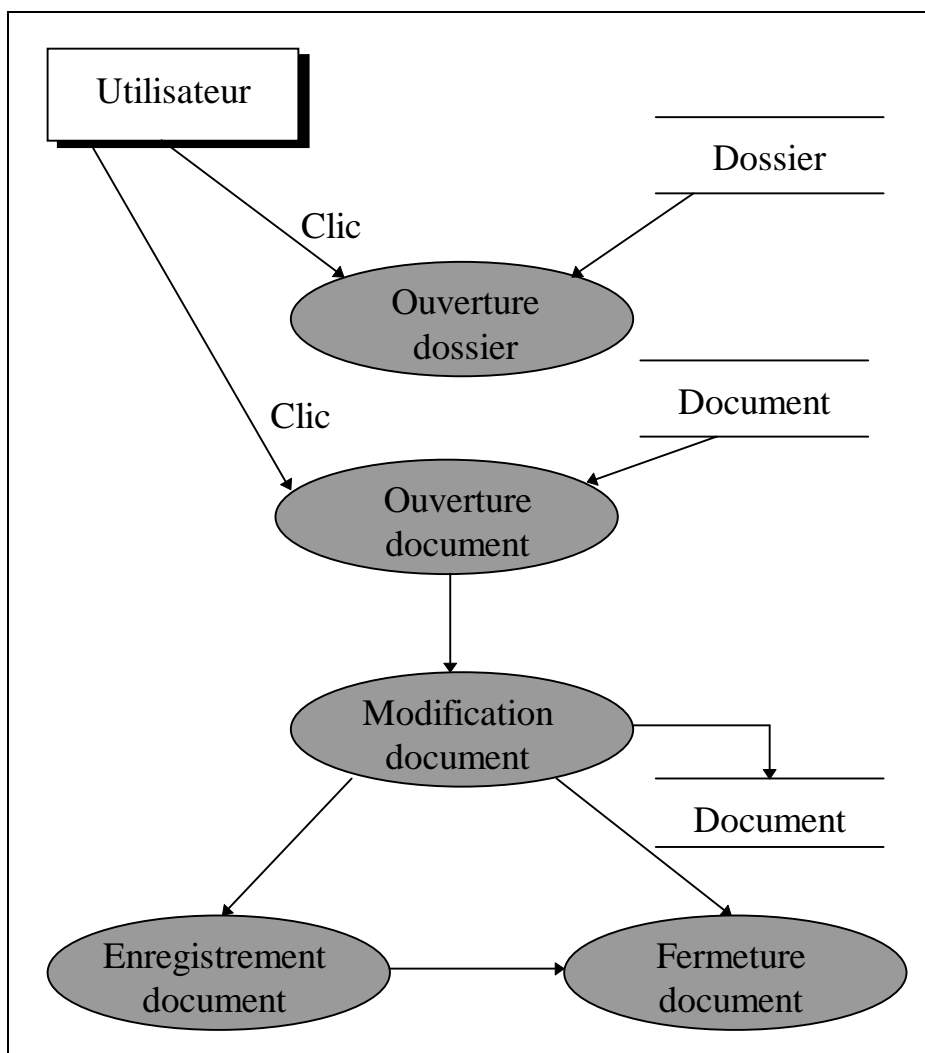


5.3.4.3 Modèle fonctionnel

Permet de décrire les **fonctions** du système. Il indique les **traitements** et les **processus fonctionnels** appliqués aux données.

Il correspond à une formalisation opérationnelle d'un scénario.

Exemple de modèle fonctionnel :



5.4 Autres méthodes objet

5.4.1 Méthode OOD

5.4.1.1 Présentation

- ◆ OOD : **Object Oriented Design**
- ◆ Développée pour le département de la défense américaine pour rationaliser le développement des applications en ADA puis en C++.
- ◆ Auteur : G. Booch
- ◆ Existe depuis le début des années 80 et a connu des versions successives.
- ◆ Méthode orientée plutôt développement : spécification technique et implémentation.
- ◆ D'autres méthodes se sont inspirées de OOD (HOOD notamment).
- ◆ Avec la méthode OOD :
 - ◇ Le modèle **statique** est développé de façon exhaustive
 - ◇ Le modèle **dynamique** est très partiellement abordé
 - ◇ L'aspect **fonctionnel** n'est pas pris en compte (OOD recommande l'utilisation de l'analyse des fonctions de la méthode SADT).

5.4.1.2 Modélisation

5.4.1.2.1 Modélisation statique

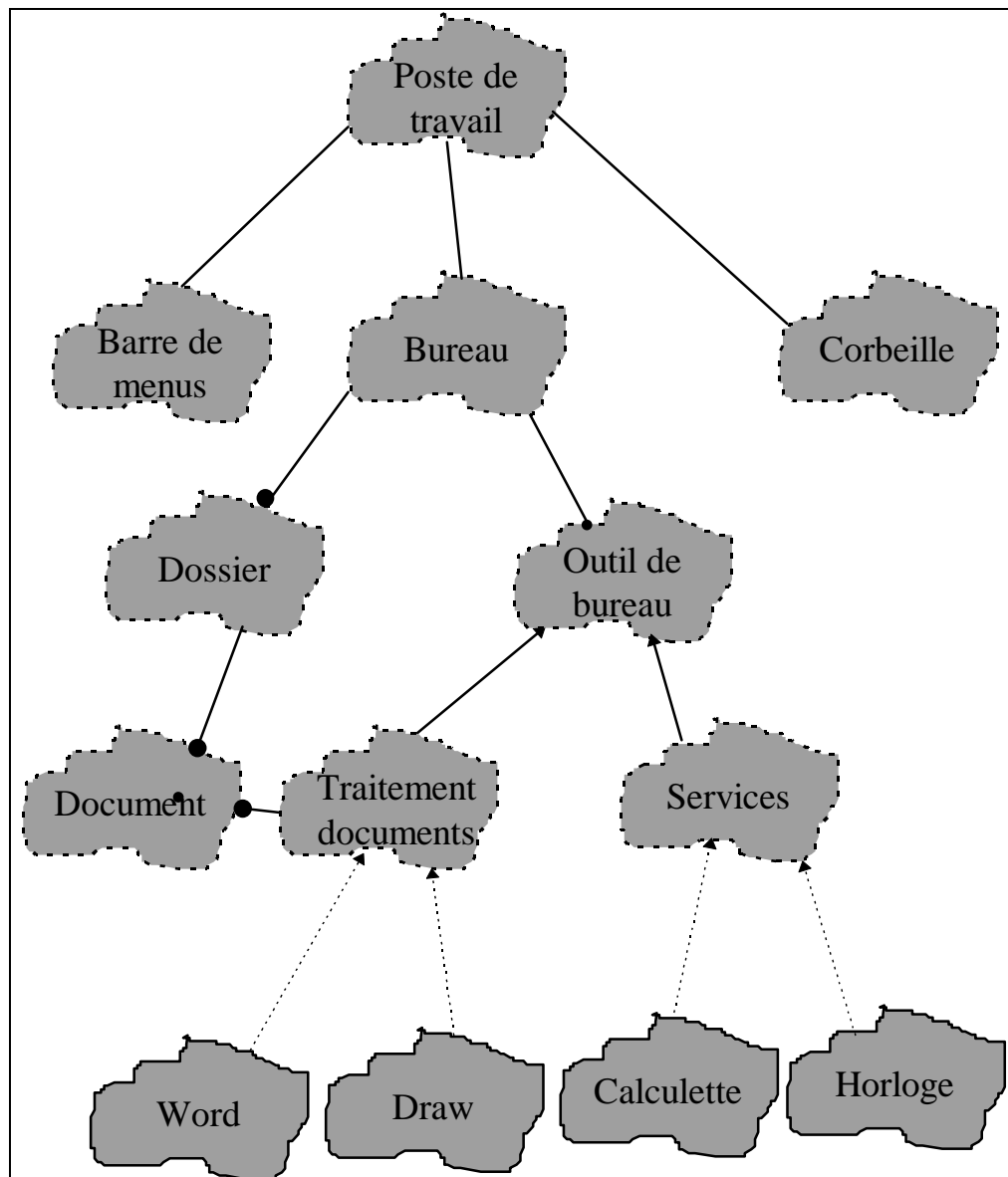
La modélisation statique OOD utilise les concepts suivants :

- ◆ **Objet** : un objet est défini par ses attributs, ses opérations, son identifiant et un ensemble d'états.
- ◆ **Association** : ce sont des liens permettant de représenter l'héritage, l'instanciation et l'utilisation des services d'un objet par un autre.
- ◆ **Objet client** : c'est un objet qui utilise les ressources d'un autre objet.
- ◆ **Protocole d'un objet** : c'est la liste des opérations qu'un objet peut effectuer sur les autres objets.
- ◆ **Comportement d'un objet** : c'est le protocole d'un objet plus la liste des opérations que les autres objets peuvent faire sur lui.
- ◆ **Rôle d'un objet** : un objet peut être :
 - ◇ *client* lorsqu'il demande des services aux autres objets
 - ◇ *serveur* lorsqu'il offre des services aux autres objets
 - ◇ *agent* lorsqu'il est à la fois client et serveur.

La modélisation statique OOD se fait à deux niveaux :

◆ **Niveau logique** : composé de deux types de diagrammes :

◇ *Diagrammes de classes* : Il décrit les hiérarchies de classes d'objets avec un certain nombre de liens entre ces classes : liens d'utilisation, liens d'instanciation et liens utilisateurs.



Le diagramme de classes est complété par une notation textuelle permettant de définir chaque classe (attributs, opérations, états, encapsulation)

◇ **Diagrammes d'objets (ou d'instances) :**
C'est la même notation appliquée aux objets. Il est utile uniquement dans le cas où les classes comportent très peu d'objets.

◆ **Niveau physique :** composé de deux types de diagrammes correspondant à l'implémentation de ceux du niveau logique. Cette implémentation est proposée en ADA, C++ et d'autres langages tels que Smalltalk et Object Pascal. :

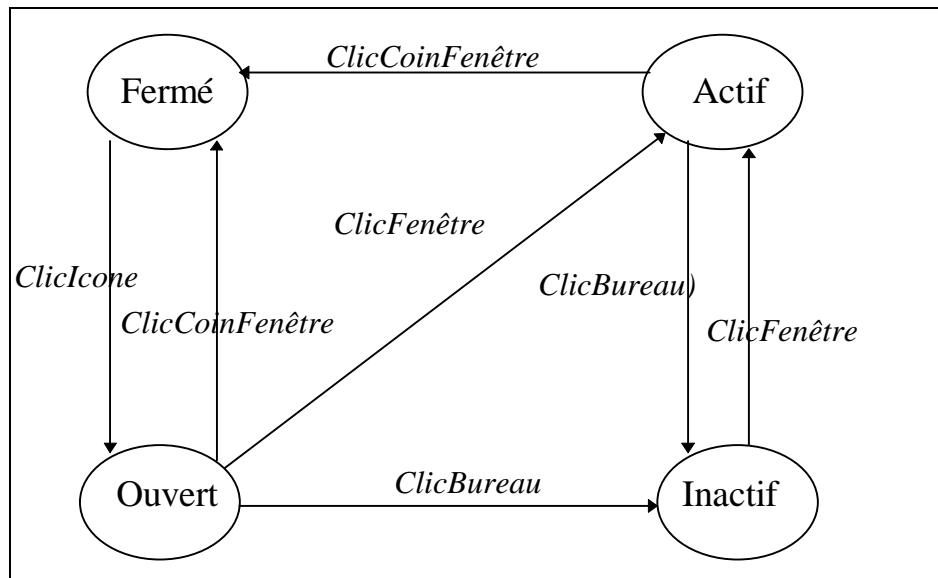
◇ **Diagrammes des modules :** Il correspond à l'implémentation du diagramme de classes.

◇ **Diagramme des processus :** Il correspond à l'implémentation du diagramme d'objets. Un processus étant une exécution particulière d'un module.

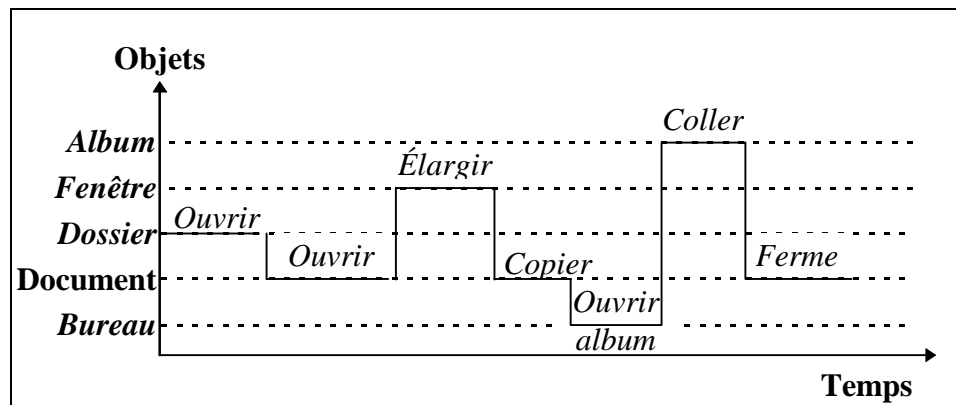
5.4.1.2.2 Modèle dynamique

OOD propose deux diagrammes pour représenter l'aspect dynamique :

◇ **Diagramme d'états/transitions** : Il représente les états d'un objet et les événements qui déclenchent les changements d'état (transitions).

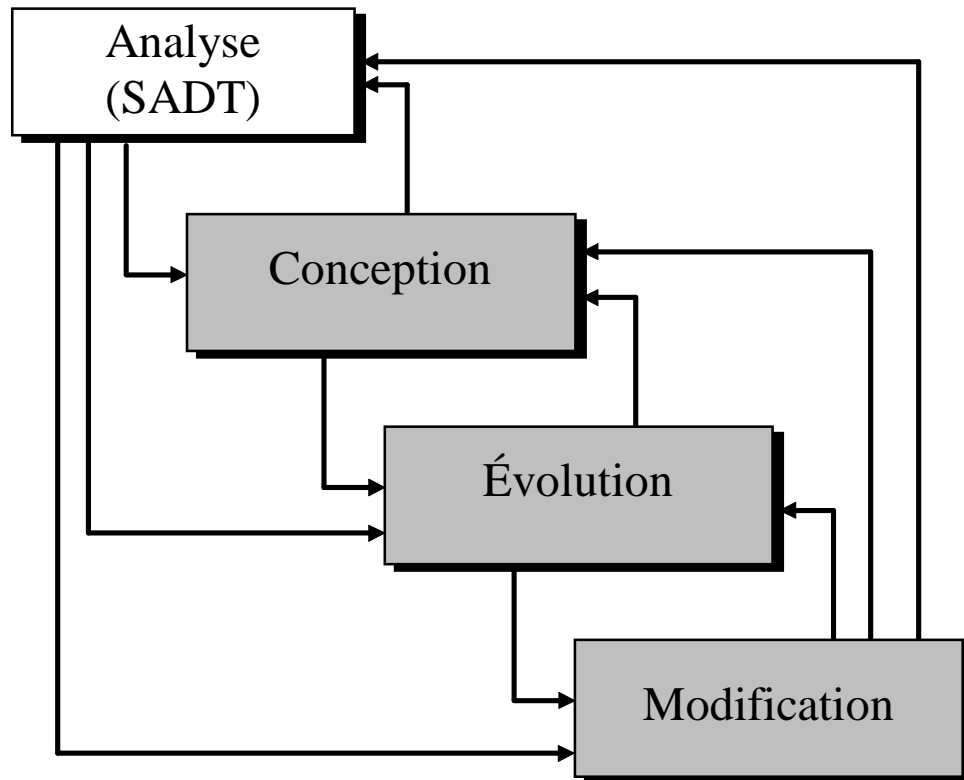


◇ **Diagramme de temps** : Il représente les interactions entre les comportements des différents objets. Il montre l'ordre d'exécution des opérations sur les différents objets.



5.4.1.3 Démarche méthodologique

OOD utilise le modèle en cascade :



Analyse : Elle n'est pas couverte par la méthode. OOD se reporte à SADT.

Conception : Elle consiste à élaborer les diagrammes de niveau logique (diagramme de classes et diagramme d'objets).

Évolution : Elle correspond au codage, test et intégration.

Modification : Elle correspond à la maintenance et aux changements pouvant survenir lors de l'évolution du système.

5.4.2 Méthode HOOD

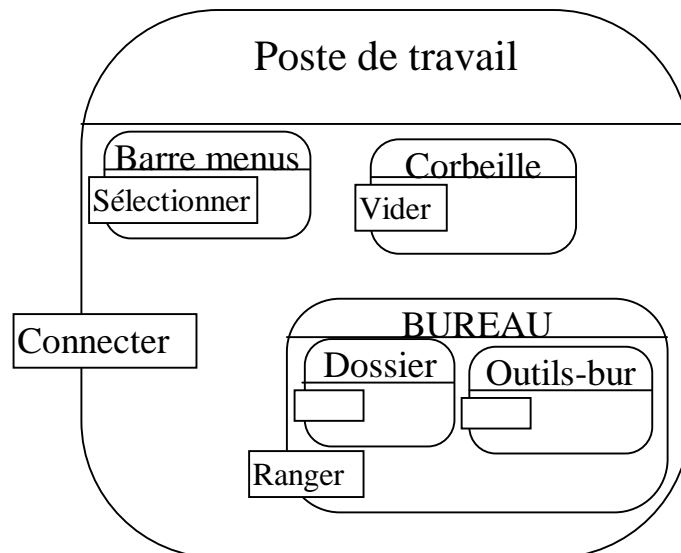
5.4.2.1 Présentation

- ◆ HOOD : **H**ierarchical **O**bject **O**riented **D**esign
- ◆ Développée par un consortium européen (CISI, Matra Espace et CRI) en réponse à un appel d'offre de l'ESA en 1987.
- ◆ Elle constitue une synthèse de OOD, des machines abstraites et des techniques de conception d'applications temps réel.
- ◆ Elle est utilisée dans de nombreux projets d'applications industrielles et aérospatiales (Colombus et Hermes).
- ◆ La méthode HOOD ne couvre que l'**aspect statique** des applications.

5.4.2.2 Modélisation statique

Principales différences par rapport à OOD :

- ◆ **Objets complexes** : HOOD introduit la notion d'objets complexes. Une relation *include* permet de composer des objets complexes à partir d'autres objets.
- ◆ Les classes sont représentées par une boîte dans laquelle peuvent être représentées les attributs et les méthodes.



- ◆ HOOD ne supporte pas l'héritage.

5.4.2.3 Démarche méthodologique

HOOD préconise une démarche par décomposition hiérarchique :

- ◆ Les objets sont vus comme des machines abstraites qui sont raffinés progressivement.
- ◆ Un objet d'un niveau d'abstraction i peut se décomposer en plusieurs objets de niveau $i+1$.

5.4.3 Méthode OOA

5.4.3.1 Présentation

- ◆ OOA : **Object Oriented Analysis**
- ◆ Développée au Lawrence Berkeley Laboratory à partir de 1979 dans le cadre d'un projet sur le temps réel.
- ◆ Auteurs : S. Shlaer et S. M. Mellors
- ◆ Orientée initialement applications temps réel (concurrency et parallélisme) puis appliquées dans d'autres domaines dont la gestion.
- ◆ La première version (1988) était consacrée à la modélisation relationnelle des données (avec quelques extensions intégrant la généralisation et l'héritage).
- ◆ La seconde version (1992) couvre les aspects dynamiques et fonctionnels.

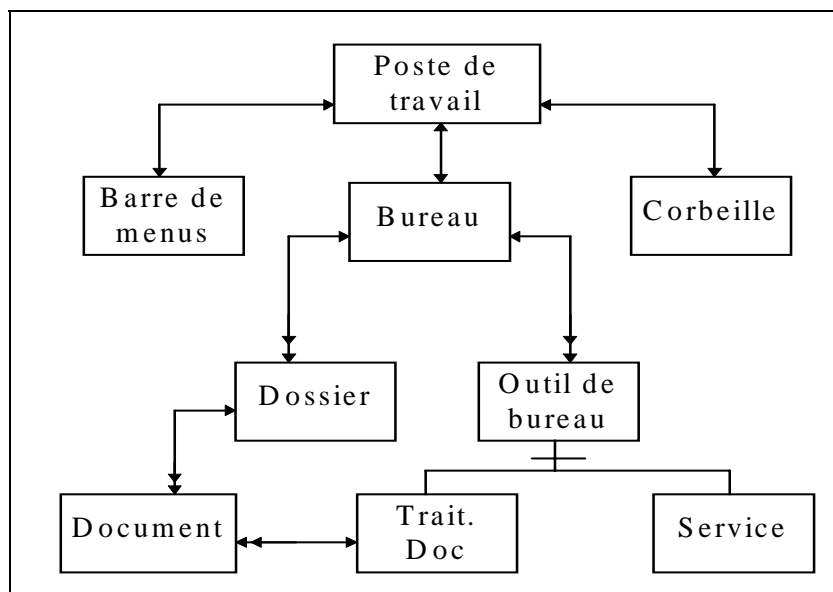
5.4.3.2 Modélisation

5.4.3.2.1 Modélisation statique

La modélisation statique OOA est basée sur un modèle dit modèle d'information, et qui est un modèle relationnel éventuellement normalisé en 3^{ème} ou 4^{ème} FN.

Ce modèle a les caractéristiques suivantes :

- ◆ La notion d'**héritage** a été rajoutée.
- ◆ La notion de **classe** et **d'objet** sont confondus et représentés par la notion d'objet.
- ◆ Les objets correspondent aux tuples.
- ◆ Les objets sont identifiés par des clés composées d'attributs.
- ◆ Il n'y a pas de méthodes attachées aux objets.
- ◆ Les associations ne peuvent être que binaires avec des cardinalités (1, 1), (1, n) et (m, n).



5.4.3.2.2 Modèle dynamique

OOA utilise un diagramme d'états/transitions pour représenter la dynamique des objets.

Chaque objet possède un **cycle de vie** décrit par une succession d'**états** dont les transitions sont déclenchées par des **événements**.

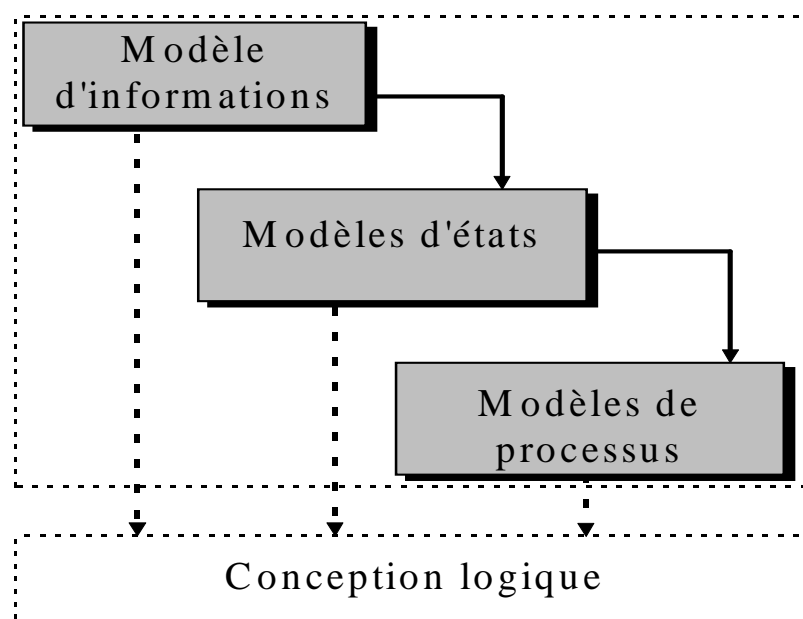
5.4.3.2.3 Modèle fonctionnel

Le modèle fonctionnel permet de spécifier le contenu de chaque activité associée à un état d'objet.

Il est supporté par un diagramme ADFD (Action Data Flow Diagram).

5.4.3.3 Démarche méthodologique

La démarche OOA constitue une mise en œuvre ordonnée des trois modèles statique, dynamique et fonctionnel.

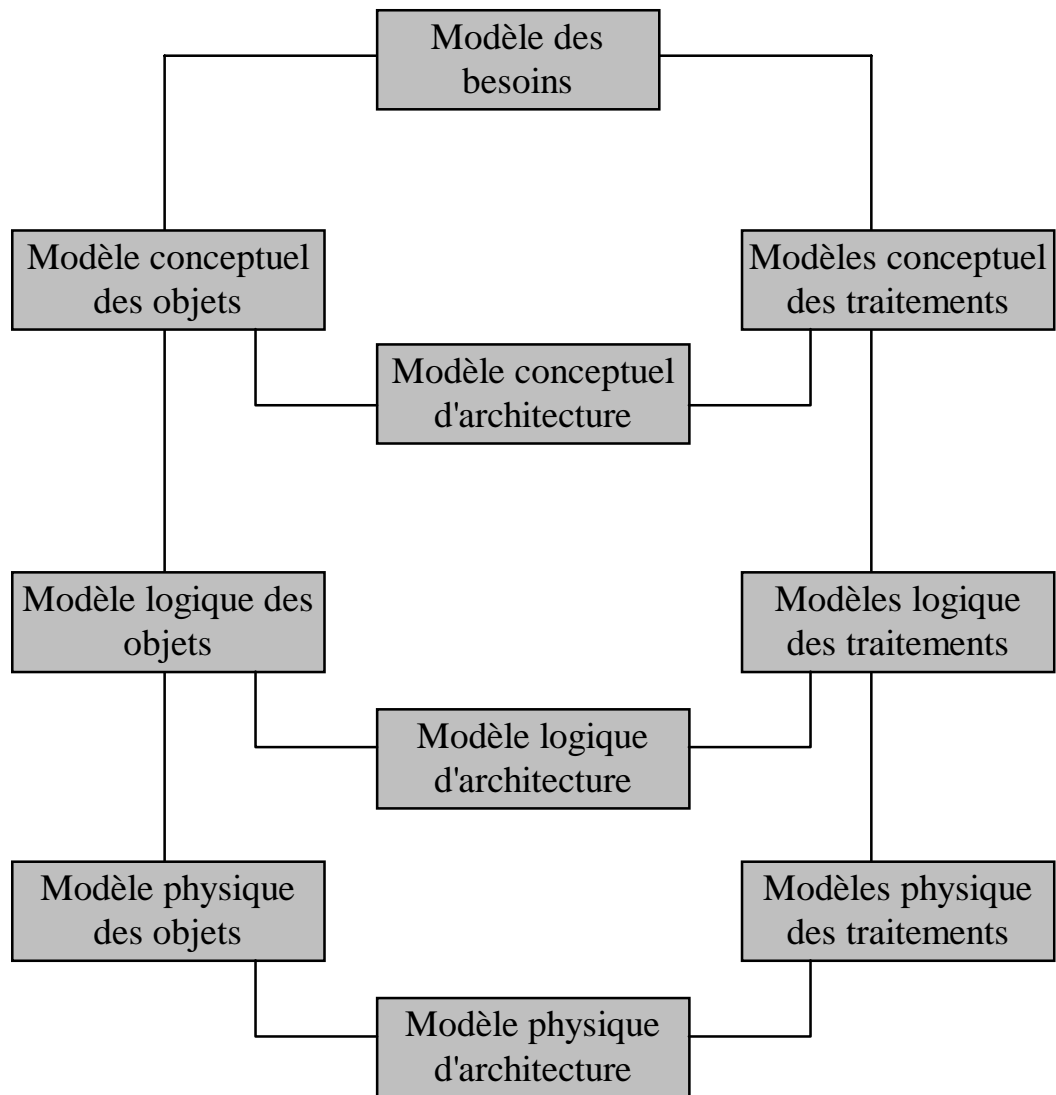


5.4.4 Méthode OOM

5.4.4.1 Présentation

- ◆ OOM : **Object Oriented Method** (Merise)
- ◆ Développée au laboratoire Prism de l'université de Versailles Saint-Quentin en France..
- ◆ Auteurs : M. Bouzeghoub et A. Rosfeld.
- ◆ OOM est une version orientée objet de Merise.
- ◆ OOM assure la compatibilité ascendante avec Merise et la conformité à l'approche objet
- ◆ OOM vise surtout à étendre la méthode de façon à ce qu'elle couvre non seulement l'analyse et la conception mais aussi la production et la maintenance.

5.4.4.2 Modélisation



Modélisation des besoins :

Au niveau des besoins, des **diagrammes de flux** sont utilisés pour représenter :

- ◆ Les acteurs
- ◆ les échanges d'informations entre les acteurs

Ces diagrammes permettent d'identifier les objets et les processus du modèle conceptuel des objets et du modèle conceptuel des traitements.

Modélisation conceptuelle :

Au niveau conceptuel, OOM utilise :

- ◆ **un modèle conceptuel des objets** : c'est un modèle entité-association étendu aux objets complexes, aux hiérarchies et au comportement dynamique. Chaque objet est défini par des **attributs**, des **opérations** et des **règles** contrôlant son intégrité ou son cycle de vie.
- ◆ **un modèle conceptuel des traitements** : C'est un ensemble de modèles correspondant aux différentes **fonctions** du SI. Chaque fonction est réalisée par un **scénario** défini sur un ensemble d'objets et elle est décrite par un but, des conditions d'exécution et une spécification formelle de sa sémantique.

Modélisation logique :

Au niveau logique, OOM utilise :

- ◆ le modèle relationnel ou le modèle objet
- ◆ les traitements sont décrits par un langage algorithmique

Modélisation physique :

Au niveau physique, OOM utilise :

- ◆ les outils d'implémentation offerts par le SGBD pour les données
- ◆ un langage de programmation standard pour la programmation des traitements.

Modélisation de l'architecture du système :

OOM permet de définir l'architecture du SI aux différents niveaux d'abstraction :

- ◆ *au niveau conceptuel* : l'architecture conceptuelle permet de définir la localisation logique des données dans les différents services, la répartition des fonctions entre ces services.
- ◆ *au niveau logique* : l'architecture logique est l'adaptation de l'architecture conceptuelle à une certaine technologie informatique (architecture centralisée, c/S, répartie).
- ◆ *au niveau physique* : l'architecture physique est l'implémentation de l'architecture logique dans un environnement matériel et logiciel donné. permet de définir la localisation logique des données dans les différents services, la répartition des fonctions entre ces services.

6. CONCLUSION ET PERSPECTIVES

◆ **Avoir une vision Système d'information :**

- ⇒ Vue globale du système d'information de l'entreprise
- ⇒ Donner la même importance aux données et aux traitements

◆ **La conception objet est une nécessité :** c'est la troisième facette de la technologie objet :

- ⇒ Programmation objet
- ⇒ Base de données objets
- ⇒ **Conception objet**

◆ **Les méthodes objets sont devenues « adultes » :**

- ⇒ unification des termes
- ⇒ unification des représentations graphiques
- ⇒ couverture des différents aspects du SI (statique, dynamique et fonctionnel)
- ⇒ fusion de méthodes (UML)

◆ Les outils d'aide à la conception constituent un facteur moteur dans la généralisation des méthodes de conception :

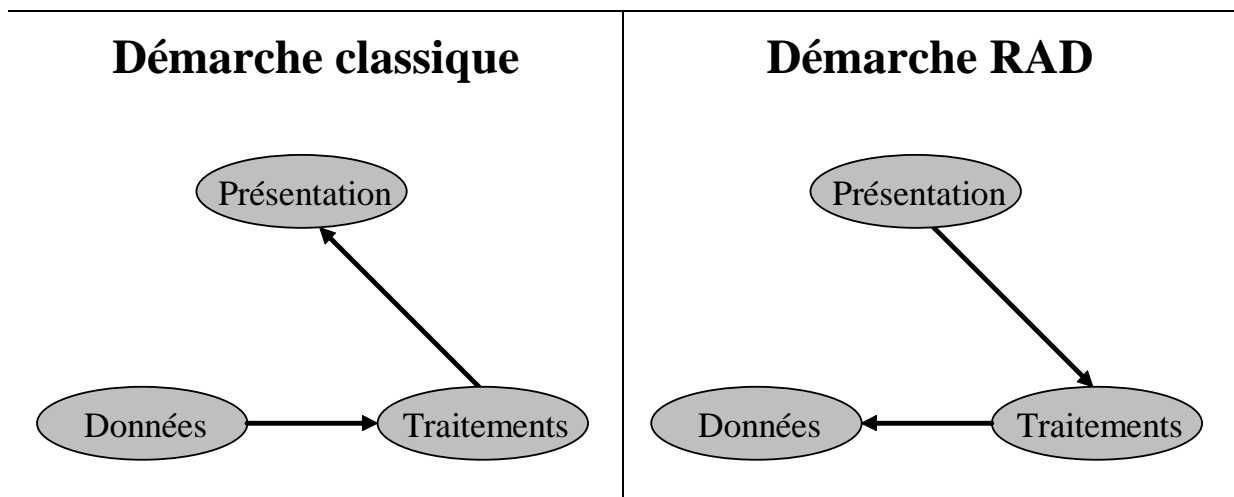
- ⇒ en couvrant toutes les composantes d'une méthode (conceptuel / logique / physique, statique / dynamique / fonctionnel).
- ⇒ en exploitant le travail d'analyse pour générer le maximum (données et traitements)

Tendance actuelle :

- ⇒ Les possibilités de génération des AGL sont en progression significative.
- ⇒ Les L4G qui proposent des outils de conception (données).
- ⇒ Une seule famille d'outils pour la conception/génération/développement ???

◆ **Démarche de conception et de développement de SI :**

- ⇒ L'approche **RAD** est mieux adaptée aux environnements techniques actuels : architecture client/serveur, interface graphique, L4G, etc.
- ⇒ Nécessité d'intégrer la composante "**Présentation**" en plus des deux autres composantes "**Données**" et "**Traitements**".



Démarche alternative

