Présentation générale de la méthode orientée objet : O.M.T. (Rumbaugh & al.)

Bernard ESPINASSE
Professeur à l'Université d'Aix-Marseille

2005

- **Introduction**
- **Les modèles d'OMT**
- **Le Modèle Objet (MO)**
- **Le Modèle Dynamique (MD)**
- **Le Modèle Fonctionnel (MF)**
- **Démarche Méthodologique d'OMT**

Les méthodes Orientées Objet

- influencées par le développement du langage **Ada** et des langages de programmation basés sur les objets **C++**
- la plupart aborde l'étude d'un problème est réalisée suivant 3 aspects :
  - aspect **statique** : identifie les propriétés des objets et leurs liaisons avec les autres objets,
  - aspect **dynamique** définit le cycle de vie des objets : **comportement** des objets, les **différents états** par lesquels ils passent et **événements** déclenchant ces changements d'états.
  - aspect **fonctionnel** : précise les fonctions réalisées par les objets par l'intermédiaire des méthodes
- **méthodes orientées objet les plus connues :**
  - **OMT** (Rumbaugh et al. 1995)
  - **OOA** (Object Oriented Analysis - Coad & Yourdon 1992)
  - **BOOCH** (Booch 1991)
  - **OOA** (Object Oriented Analysis - Shlaer & Mellor 1992)
  - **Objectory-OOSE** (Jacobson & al. 19XX)
  - **HOOD, …**

O.M.T.

- **OMT** pour **Object Modeling Technique :** actuellement connaît un grand succès tant en Amérique du Nord qu'en Europe
- à l'origine développée au sein de **General Electric** à la fin des années 80, par **Rumbaugh**, **Blaha, Premerlani, Eddy** et **Lorensen**
- constante évolution :
  - premier rapprochement avec la méthode OOD (Object Oriented Design) de G.Booch [Booch 87] et
  - récent rapprochement avec la méthode suédoise OOSE (Object Oriented Software Engineering - Objectory - de Jacobson et al. (1992)
  - U.M.L. : Unified Modeling Language ....
- **bibliographie complémentaire :**

  - Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., "Object-Oriented Modeling and Design", Prentice Hall, 1991 - traduction française : "OMT : Modélisation et conception orientées objet", Masson - Prentice Hall, 1994

    (tome 1 et 2)

Les modèles d'OMT

- comme la plupart des méthodes orientées objet, 3 dimensions de modélisation : objet, dynamique et fonctionnelle :
  - **Modèle Objet (MO)** : identifie les propriétés des objets et leurs liaisons avec les autres objets,
  - **Modèle Dynamique (MD)** : définit le cycle de vie des objets en précisant :
    - le **comportement** des objets,
    - les **différents états** par lesquels ils passent et
    - les **événements** qui déclenchent ces changements d'états.
  - **Modèle Fonctionnel (MF)**: précise les fonctions réalisées par les objets par l'intermédiaire des méthodes

Nous présenterons de façon succincte les 2 aspects principaux de la méthode :

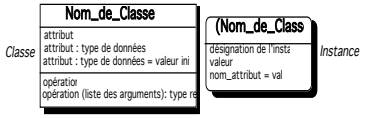
- les **3 modèles proposés**
- la **démarche générale préconisée**.

Le modèle objet d'OMT (MO)

- modèle **le plus important**
- **extension du modèle Entité-Association** par l'introduction de :
  - l'**agrégation**,
  - la **généralisation**
  - la spécification d'**opérations** et **contraintes** au niveau des entités, nommées ici "**Classes**"

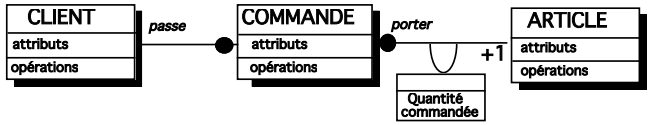
Classes et Instances

- une classe d'objets modélise un **ensemble d'objets** ayant des **propriétés similaires** (Attributs), des **comportements similaires** (Opérations)
- le modèle de classes présente ces **classes**, leurs **hiérarchies** et leurs **relations** entre elles.
- modélisation d'**instances** de ces classes permettant d'élaborer des diagrammes d'instances exprimant des **scénarios**
- **conventions** graphiques adoptées :

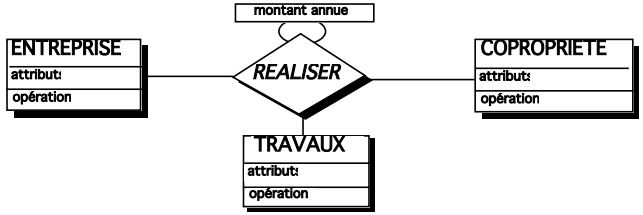


Associations binaires et n-aires

- des associations **peuvent relier des classes non nécessairement distinctes**
- 2 types d'associations :
  - associations **binaires**, représentées par un lien :



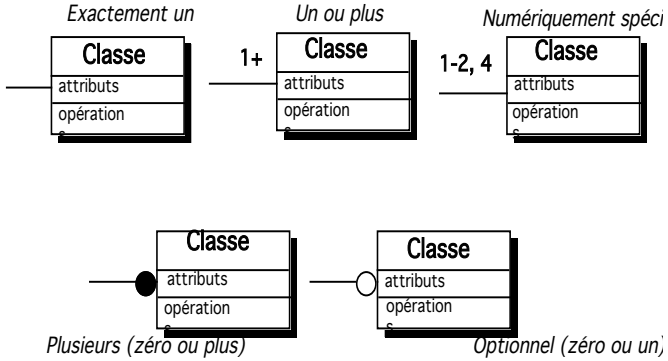
  - associations **n-aires** représentées par un **symbole spécifique** :



- à ces associations peuvent être rattachés des **attributs** et/ou des **opérations** propres

Multiplicités

- caractérisent les associations **binaires**
- spécification des multiplicités à l'autre bout de la ligne (à l'inverse des cardinalités Merise).

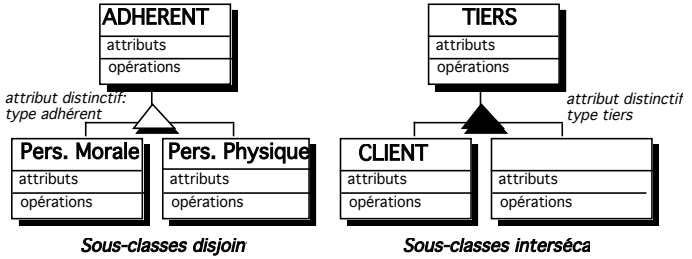


Remarque :

la notion de multiplicité **ne fonctionne plus** avec les associations **n-aires**.

Généralisation

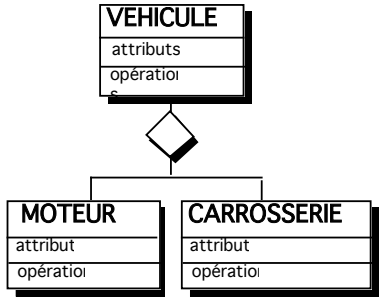
- relation **transitive** entre une classe et une ou plusieurs autres classes, nommées sous-classes, qui sont des raffinements de la première, nommée **surclasse**
- peut être basée sur un attribut spécifique, nommé **attribut distinctif**



- chaque sous-classe **hérite** des caractéristiques (attributs et opérations) de la surclasse
- toute sous-classe peut modifier et/ou ajouter de **nouveaux attributs** ou **opérations** par rapport à ceux hérités
- possibilité de **contraintes** sur les hiérarchies de classes (contraintes **d'intersection/disjonction)**
- sous-classes **disjointes** : intersection vide
- sous-classes **intersécantes** : intersection non vide.

Agrégation

• relation **transitive** du type Composé-Composant entre classes :

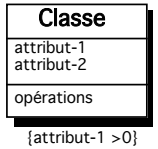


une classe Véhicule est composée par exemple de 2 autres classes, les classes Moteur et Carrosserie

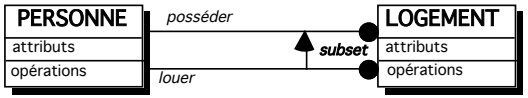
Contraintes

• **restreignent** le champ des **valeurs possibles** des éléments sur lesquels elles portent :
  • **objet**
  • **classe**
  • **attribut**
  • **liaison**
  • **association**

• **contraintes d'attributs** : restreignent valeurs pouvant être prises par un attribut d'une classe:



• **contraintes inter-associations** : enrichissent la sémantique en permettant par exemple l'expression de contrainte **d'inclusion - subset** :



Clés candidate

• clé candidate = **ensemble minimum d'attribut identifiant** de manière unique un objet ou une association

• **indispensable** dans le cas d'**association n-binaire**

Autres possibilités **dans le Modèle Objet d'OMT :**

• **Meta-classes** : classes permettant de décrire d'autres classes comme des instances;

• **Classes abstraites** et **classes concrètes** : classe sans instance mais dont les descendants, nommés classes concrètes, ont des instances;

• **Classes jointe** : classe associée à plusieurs surclasse (héritage multiple);

• **Classes dérivées** : classe dont les instances peuvent êtres calculées à partir d'autres classes;

• **Associations dérivées** : association dont les instances peuvent êtres calculées à partir d'autres associations;

Conclusion sur le Modèle Objet d'OMT

• est une **extension à l'objet du modèle Entité-Association** par adjonction d'opérations et de contraintes au concept d'entité

• possède une **puissance d'expression importante**

• assez **voisin du formalisme Entité-Relation** proposé par **MERISE** 2ème génération : couplage aisé

• certaines **notions moins avancées et moins clairement formalisées** que dans MERISE :

  • identification des relations,

  • cardinalités,

  • contraintes d'intégrité fonctionnelles,

  • contraintes inter-relations,

  • nature des attributs (simple ou complexe, monovalué ou multivalué),

  • unicité de modélisation des relations qu'elles soient binaire, n-aires ou porteuses de propriétés)

Le modèle dynamique d'OMT (MD)

- représentation des **aspects de l'application affectés par le temps** et de **spécifier l'ordonnancement de l'exécution des opérations** déjà définies dans le modèle objet
- une opération se produit sans prendre en compte ni son action, ni l'objet sur lequel elle agit, ni la manière dont elle est accomplie
- c'est un **diagramme d'états-transitions** spécifiant l'ordonnancement des **états** et des **événements** autorisés pour une classe d'objets :

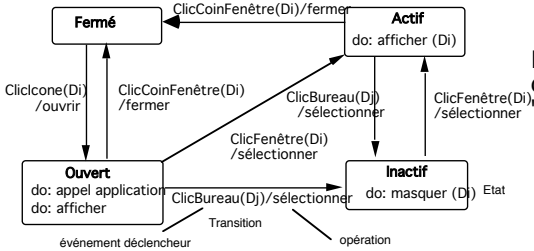


Diagramme d'états de l'objet "Document"

- sont mentionnés les **événements déclencheurs** des **transitions** et les **opérations** de transformation associées: **cycle de vie des objets**
- les **événements** de ce diagramme correspondent aux **opérations** définies dans les classes et associations du MO

L'événement

événement = l'unique porteur d'information d'un objet vers un autre objet

- **sans durée** et un attribut **date** le caractérise implicitement
- un **objet** qui **envoie** un **événement** vers un autre **objet** pourra attendre une **réponse** qui sera elle-même **un autre événement**
- consiste en un simple signal ou peut avoir des **attributs d'événement** définissant le message qu'il transporte
- **regroupement d'événements** de même structure de message et de comportement commun en **classes d'événements.**
- la structure des classes d'événements similaire à celle des classes d'objet du MO
- peuvent s'enchaîner par des liens de causalité ou survenir de façon concurrente.

Etat

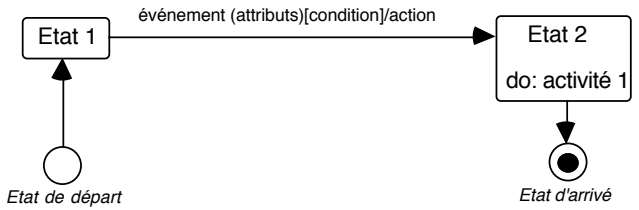
- **état** = valeur d'un objet durant un **intervalle de temps** ou durant l'occurrence de **2 événements**
- à chaque **état remarquable**, on associe une **description textuelle** du type :

```
Etat: <nom_d'état>
Description: <string>
Séquence d'événements qui produit
l'état:
Evénement -1
Evénement -2 ...
Conditions qui caractérisent l'état:
Condition -1
Condition -2 ...
Evénements qui sont acceptés par
l'objet dans cet état:
Evénement Action
Prochain état:
...
```

- durant cet intervalle de temps, **l'objet peut réaliser une action qui ne change pas son état de façon significative**
- le passage d'un état à un autre se traduit par la **modification de la valeur d'un ou de plusieurs attributs de l'objet concerné**

Transition

- décrite par **l'événement** qui la déclenche, les **attributs** qui caractérisent cet événement, la **condition** à vérifier en plus de l'occurrence d'événement et l'action à exécuter



- l'action que doit exécuter un objet dans un certain état introduite par le mot-clé **"do: "**
- pour **chaque classe d'objets** le nécessitant, on développera un **diagramme d'états** représentant son **cycle de vie**
- **les diagrammes d'états de l'ensemble des classes constituent le modèle dynamique**

Exemple de diagramme d'état associé à un objet

Diagramme d'état associé à un objet *Document*. (Di) :

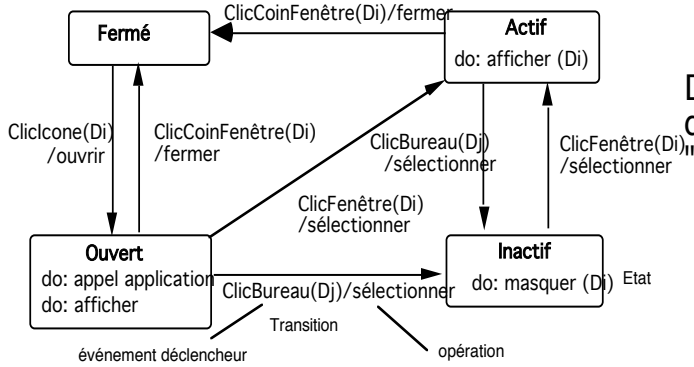


Diagramme d'états de l'objet "Document"

- les principaux événements sont ici des **clics de souris** réalisés par **l'utilisateur**
- les **actions** accompagnants chacun des événements (ouvrir, fermer, sélectionner) sont considérées comme **atomiques**
- les activités ***afficher*** et ***masquer*** introduites par le mot-clé **do:** sont des **activités permanentes**

Scénario

- permet d'identifier les **événements** et à **réaliser les diagrammes d'états**
- se définie comme une **séquence d'événements** qui se produit lors d'une **exécution** du système

est au modèle dynamique ce que le modèle d'instance est au modèle objet

- représenté par une **liste ordonnée** de nom d'événements, par exemple :

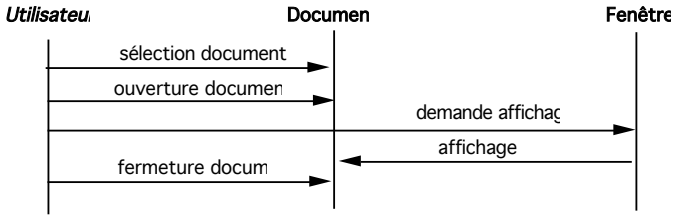
{événement-1, événement-2, événement-3, événement-4, événement-5, événement-6}

- permet **d'identifier**, pour chaque **classe d'objets**, les **événements** qui **arrivent** ou sont **émis** par cette **classe**
- **l'ensemble des scénarios décrit l'activité du système**, il représente en quelque sorte les **fonctions** du système

Traceurs d'événement

- diagramme représentant la **séquence d'événement associée à un scénario mais aussi les objets** qui **envoient** et **reçoivent** ces **événements**

exemple :

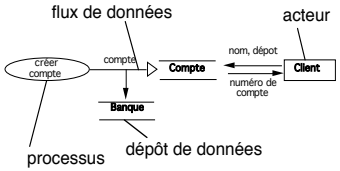


traceur associé aux objets **Document** et **Fenêtre** :

- *c'est à partir de ces **événements arrivant** ou **partant** que l'on peut élaborer le **diagramme d'états d'une classe**.*

Le modèle fonctionnel d'OMT (MF)

- modélise les **processus de transformation** de l'application, les transformations des valeurs-fonctions, "mapping", contraintes et dépendances fonctionnelles
- **constitue une formalisation opérationnelle d'un scénario**
- spécifie la signification des **opérations** et **contraintes** du MO et les **actions** du MD
- **le lien entre ce MD et le MF** repose sur les **actions** qui représentent les **fonctions** du MF
- c'est un **diagramme de flux** classique modélisant :
  - des **processus** transformant des données : **ellipse**
  - des **flux de données** transportant des données : **flèche** (si génère un objet se terminera par un triangle blanc, sinon noir)
  - des **objets acteurs** produisant et consommant des données : **rectangle**
  - des **dépôts de données** qui stockent ces données.



Processus

- **transforme des données**
- peut être **contrôlé** par une valeur booléenne appelée **flot de contrôle**, montrant ainsi par exemple la dépendance entre processus (**flèche** en trait pointillé partant du processus contrôlant vers le processus contrôlé)

tout processus met en oeuvre 4 grands types d'opérations :

1 • **opérations d'accès** : écrivent ou lisent des attributs ou des liaisons de l'objet auxquelles elles sont rattachées;

2 • **opérations requête** : restreintes aux états visibles d'un objet, c'est à dire sans influence sur les autres objets;

3 • **opérations action** :

- **produisent des effets sur des autres objets modifiant leurs états**
- **sont dérivées des processus du MF** : peuvent être spécifiées par des formules arithmétiques, des tables ou arbres de décision, algorithmes, texte en langage naturel
- **sont instantanées et dont de durée nulle;**

4 • **opérations activité** : produisent des effets sur des autres objets modifiant leurs états et ont une durée dans le temps;

Flux de données, Acteur et Dépôt de données

Flux :

- assure **connexion** entre la sortie d'un objet ou d'un processus et l'entrée d'un autre processus
- il représente une valeur intermédiaire d'une données qui peut être une donnée simple ou un agrégat de données

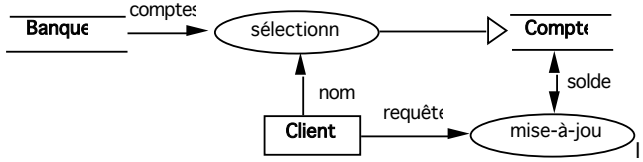
Acteur : **objet actif** comportant une **opération stimulant** un **processus** ou étant **concerné en sortie d'un processus**

Dépôt de données :

- objet passif assurant le **stockage de valeurs** pouvant être simples ou agrégées (listes, tables), en vue d'une utilisation ultérieure
- les **flèches y entrant** représente les **opérations** ou les **informations modifiant** les données mémorisées dans le dépôt,
- **les flèches en sortant** représentant les **informations demandées** au dépôt.

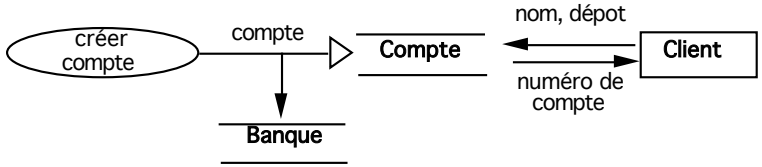
Exemples de modèles fonctionnels

1) **modèle fonctionnel de sélection** avec un **objet** comme **résultat** :



e nom du client "sélectionne" un compte de la banque, le résultat de cette sélection est l'objet Compte lui-même qui est alors utilisé comme dépôt de données dans l'opération de "mise à jour"

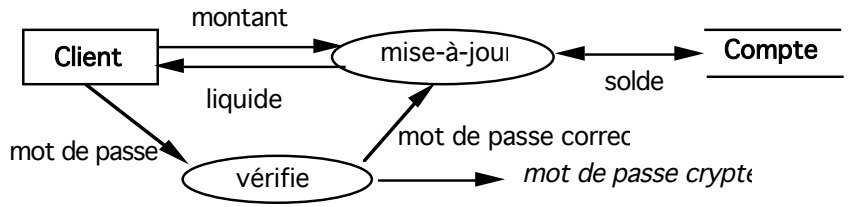
2) modèle fonctionnel de création d'un nouvel objet :



la création d'un nouveau compte dans une banque. Le résultat du traitement "créer compte" est un nouveau compte stocké dans le dépôt de données Banque. Le numéro de compte est transmis au client.

Exemples de modèles fonctionnels

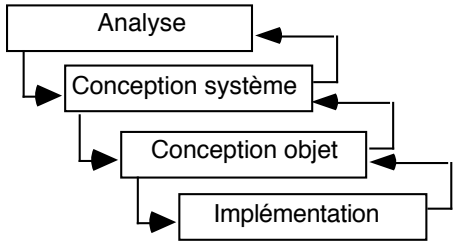
3) modèle fonctionnel avec un flot de contrôle :



pour un retrait sur son compte en banque, le client fournit un mot de passe et un montant, le retrait n'est possible que si le mot de passe est correct. Le flot de contrôle sur le processus Vérifier est ici le mot de passe crypté connu du système.

Démarche Méthodologique d'OMT

- cycle de développement d'OMT **du type en cascade**
- composé des phases :
  - d'**Analyse**
  - de **Conception du système**
  - de **Conception des objet**
  - d'**Implémentation**
- seules les **itérations** vers les **phases immédiatement précédentes** sont **permises**



Démarche Méthodologique d'OMT

- **Analyse :**

développer un **modèle de ce que le système** doit faire **sans se préoccuper de la façon avec laquelle il sera Implémenté**

- **Conception système :**

consiste en une **optimisation**, un **affinage** et une **extension** des **MO, MD, MF,** de façon suffisamment **détaillée** pour une **implémentation**

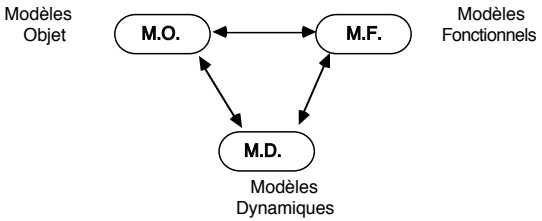
- **Conception des objets :**

intègre le point de vue technique de l'implémentation concrète

La phase d'Analyse

Objectif : **décrire ce que le système à concevoir "doit" faire** et non "comment" le faire

- élaboration **progressive** et **itérative** des 3 modèles (MO, MD, MF) du système :



- on commence de façon un peu intuitive par le **MO** en faisant émerger les classes essentielles,
- puis on élabore les **MD** et **MF**,
- **revenir** ensuite sur le **MO**, afin de **l'affiner** en introduisant par exemple de nouveaux attributs, **opérations** et **contraintes**.

La phase d'Analyse : Élaboration du Modèle Objet

- le **MO** favorise la **communication** entre informaticiens et les experts du domaine
- l'élaboration du MO n'est **pas un processus linéaire**, de nombreux allers-retours avec les 2 autres modèles sont nécessaires

Étapes préconisées :

- **identifier les classes d'objets;**
- **constituer (itérativement) un dictionnaire de données** (contenant les descriptions des classes, attributs et associations);
- **ajouter les associations entre les classes d'objets;**
- **ajouter les attributs des classes d'objets et des associations;**
- **organiser et simplifier les classes d'objets en utilisant l'héritage;**
- **vérifier les chemins d'accès en utilisant les scénarios;**
- **grouper les classes dans des modules** (couplage fort, fonction apparentées).
- faire une **itération** sur le modèle pour le raffiner;

La phase d'Analyse : Élaboration des Modèles Dynamiques

- le **MD** spécifie les comportements dans le temps du système et des objets le représentant :

Quand ?

- on recherche les événements pertinents associés à ces comportements, ainsi que les séquences d'événements valides ou scénarios pour chaque objet que l'on modélisera dans un diagramme d'états

Étapes préconisées :

- **préparer les scénarios des séquences d'interaction typiques**;
- **identifier les événements entre objets**;
- **préparer un traceur d'événements pour chaque scénario;**
- **construire un diagramme d'états pour chaque classe** ayant un comportement dynamique important;
- **vérifier la cohérence et la complétude des événements partagés parmi tous les diagrammes**

La phase d'Analyse : Élaboration du Modèle Fonctionnel

- le **MF** spécifie **comment** les **valeurs** sont **calculées** indépendamment de l'ordonnancement, des décisions ou de la structure des objets et quelles **dépendent** les unes des autres, ainsi que les **fonctions** qui relient ces valeurs
- un processus d'un DFD correspond à des activités ou actions du diagramme d'états
- un flux du DFD correspond à des objets ou à des valeurs d'attributs d'un diagramme d'objets

=> élaborer le modèle fonctionnel après l'élaboration des 2 autres modèles

Étapes préconisées :

- **identifier les valeurs d'entrée et de sortie;**
- **construire les DFD** là ou c'est nécessaire pour mettre en évidence les dépendances fonctionnelles;
- **décrire ce que fait chaque fonction;**
- **identifier les contraintes;**
- **spécifier des critères d'optimisation;**

La phase d'Analyse : Élaboration du Modèle Fonctionnel

- les 3 modèles construits, il s'agit **itérativement de vérifier chaque modèle, le faire interagir avec les 2 autres**
- enfin les **valider avec les experts du domaine d'application**

Étapes préconisées :

- **ajouter un MO les opérations-clés pendant la préparation du MF**
- **vérifier que les classes, associations, attributs et opération soient cohérents et complets;**
- **développer des scénarios plus élaborés (avec les conditions d'erreurs) comme variations des scénarios de base;**
- **itérer;**

La phase Conception Système

- définir **l'architecture globale**, les **constituants** et les **ressources** du système informatique, "**comment**" le système **doit faire**
- **décomposer** le système en **sous-systèmes**, identifier des conflits éventuels, de choisir les stratégies de stockages, d'accès aux données et de contrôles à effectuer

Étapes préconisées :

- **organiser le système en sous-systèmes;**
- **identifier les concurrences inhérentes au problème;**
- **soumettre les sous-systèmes aux processeurs et aux tâches**
- **choisir la stratégie de base pour l'implémentation des dépôts de données** : structures de données, fichiers et BdD
- **identifier les ressources globales (volumétries) et déterminer les mécanismes pour contrôler leurs accès**
- **identifier les conditions critiques;**
- **faire des choix entre les priorités.**

La phase Conception Objet (1)

Étapes préconisées :

- **obtenir les opérations pour le MO :**
  - trouver une opération pour chaque traitement du MF
  - définir une opération pour chaque événement du MD, en fonction du contrôle
- **concevoir les algorithmes pour implémenter les opérations :**
  - choisir les algorithmes qui minimisent le coût d'implémentation des opérations
  - sélectionner les structures de données appropriées à ces algorithmes
  - redéfinir (si nécessaire) de nouvelles classes et de nouvelles opérations
- **optimiser les chemin d'accès aux données :**
  - réorganiser les calculs pour plus d'efficacité,
  - enregistrer les résultats de calculs pour éviter des calculs pour éviter des calculs redondants
  - minimiser les coûts des chemin d'accès
- **implémenter le contrôle du logiciels.**