MASTER EN SCIENCES INFORMATIQUES

Contribution à l'élaboration d'un méta-modèle pour les méthodes de conception de systèmes d'information

Mouchet, Philippe

Award date:
1991

Awarding institution:
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

CONTRIBUTION A L'ELABORATION D'UN META-MODELE POUR LES METHODES DE CONCEPTION DE SYSTEMES D'INFORMATION

Philippe MOUCHET

Mémoire réalisé sous la direction des
Professeurs J. RAMAEKERS & E. DUBOIS

présenté en vue de l'obtention du titre de
Licencié et Maître en Informatique

Année académique 1990-1991

Résumé

Sur base des modèles proposés dans diverses méthodes de conception de système d'information, nous allons élaborer un méta-modèle reprenant les concepts destinés à la modélisation des aspects tant statiques que dynamiques.

Ce méta-modèle est destiné à servir de base à l'élaboration du "dictionnaire" d'une interface entre outils de conception et outils de réalisation, telle que le "AD/Cycle Repository" proposé par IBM, ou le programme "LINC CASE INTERFACE" d'UNISYS.

Les méthodes abordées sont MERISE, INTERACTIVE DESIGN APPROACH (IDA), REMORA, OBJECT ORIENTED ANALYSIS (OOA), et STRUCTURED ANALYSIS & DESIGN TECHNIQUES (SADT).

Abstract

On the basis of models proposed in methods for modeling information systems, we will elaborate a meta-model presenting a taxonomy of the concepts used to represent statics and dynamics aspects.

This meta-model is planned to be used in the context of a software interface repository between requirements modelling and implementation tools, like the IBM "AD/Cycle Repository", and the UNISYS "LINC CASE INTERFACE" program.

The analysed methods are MERISE, INTERACTIVE DESIGN APPROACH (IDA), REMORA, OBJECT ORIENTED ANALYSIS (OOA), and STRUCTURED ANALYSIS & DESIGN TECHNIQUES (SADT).

Je tiens à remercier Messieurs les Professeurs J. RAMAEKERS et E. DUBOIS, pour avoir accepté de promouvoir ce mémoire et pour m'avoir conseillé efficacement tout au long de son élaboration.

Je tiens à exprimer ma profonde reconnaissance à toutes les personnes qui m'ont permis de réaliser ce mémoire dans les meilleures conditions.

Je remercie tout particulièrement:

Messieurs Ph. GEORGET et M. COULON pour leur accueil au sein de la société UNISYS, à Val-de-Reuil,

Tous les membres de l'équipe LCI du Val-de-Reuil, et en particulier Monsieur A. BOUTIN pour ses conseils judicieux et amicaux.

Enfin, je tiens à remercier tous ceux et celles qui, tout au long de mes études, m'ont encouragé et soutenu.

Table des matières

0. Introduction

0.1. Les systèmes d'information

Faisant face à des volumes d'informations de plus en plus grands, et afin de les gérer de la manière la plus efficace possible, les organisations ont été amenées à développer des "systèmes d'information".

A l'origine, vu le nombre restreint d'informations à gérer et l'absence de moyen technique efficace, les organisations avaient recours aux fiches. Avec l'avènement de l'informatique, ces fiches furent peu à peu remplacées par des médias plus modernes dont l'ordinateur et les fichiers informatiques sont les bases.

De l'informatique de gestion comptable, nous sommes maintenant passé à l'informatique de gestion des informations, aux systèmes d'informations.

Il existe divers types de systèmes d'information:

• Les systèmes destinés à améliorer les procédés, en vue de les rendre plus rapides, plus fiables, et moins couteux. C'est le cas par exemple des systèmes automatisés de gestion de stock.

• Les systèmes destinés à aider aux prises de décision stratégiques et non plus fonctionnelles de l'organisation.

Nous ne nous intéresserons qu'aux systèmes correspondant à la première catégorie.

0.1.1. Définition

Un système d'information (S.I.) est la combinaison de quatre ensembles [BOD. 89], [ROL. 88]: un ensemble de données, un ensemble de traitements, un ensemble de règles d'organisation, et un ensemble de ressources.

Un ensemble de données.

Les données représentent les informations et les faits significatifs pour l'organisation, que l'on désire conserver en vue de les exploiter. On ne représente que les informations jugées pertinentes.

Il peut s'agir d'informations sur les clients (nom, adresse), les produits et les stocks, sur les modes de production, ou encore d'informations concernant l'évolution de l'entreprise (chiffre d'affaire, incidents, ...).

Un ensemble de traitements

Ces traitements permettent de manipuler les données conservées. Il s'agit de procédés permettant d'acquérir et de mémoriser des données, de les sélectionner, les transformer, et enfin les restituer ou les communiquer.

Un ensemble de règles organisationnelles

Ces règles fixent les modalités d'utilisation des traitements, et notamment l'interprétation des données. Elles sont directement déduites des procédures de fonctionnement de l'organisation.

Un ensemble de ressources

Il s'agit des moyens techniques et humains requis pour l'application des règles organisationnelles et l'exécution des traitements.

0.1.2. Système d'information et organisation

Un système d'information est donc une modélisation de la réalité de l'organisation. Il contient une représentation d'aspects statiques (informations) et dynamiques (règles et traitements) de l'organisation. Il permet d'appréhender la réalité organisationnelle de manière simplificatrice.

En plus de représenter l'organisation, il en fait partie. En effet, le système d'information créé pour simplifier certains processus organisationnels va devenir un outil, une ressource pour cette organisation.

Un système d'information est dès lors indissociable de l'organisation qu'il représente.

0.1.3. Cycle de vie d'un système d'information

Lorsque la décision de créer un système d'information est prise, on s'accorde généralement pour dire qu'il va suivre le cycle de vie composé des trois phases suivantes: phase de conception, phase de réalisation, phase d'utilisation et d'évolution.

Phase de conception

La phase de conception, c'est-à-dire de modélisation de l'organisation, est primordiale. Il s'agit, en effet, de créer une représentation abstraite de l'organisation, de ses activités et des relations qu'elle entretient avec son environnement.

On va donc devoir modéliser les aspects statiques, c'est-à-dire les données, et les aspects dynamiques (les traitements). Selon l'approche générale que l'on aura du système, on modélisera données et traitements séparément, pour ensuite établir des liens, ou conjointement, sans jamais les dissocier.

La modélisation sera basée sur l'expression, dans un formalisme déterminé, des faits et règles régissant l'organisation. On élaborera les spécifications fonctionnelles du système.

Cette phase est celle demandant le travail le plus créatif. On va rechercher la solution la plus appropriée à l'organisation, indépendamment de toute entrave technique, due aux limitations technologiques.

Cette phase est la plus importante car toute erreur commise lors de la modélisation, engendrera la création d'un système non conforme et inadapté à l'organisation.

En plus, les besoins, sans cesse changeants, des organisations imposent une modélisation suffisamment souple et adaptable pour pouvoir suivre ces changements..

Le résultat de cette phase devra donc être complet, cohérent, précis, aisément compréhensible, conforme au modèle de référence (l'organisation) et aux besoins exprimés.

Phase de réalisation

Il s'agit, sur base du modèle de l'organisation fourni par la première phase, de créer les fichiers informatiques et les programmes nécessaires au bon fonctionnement du système.

Il est évident que ce travail sera d'autant facilité que la modélisation du système aura été faite de manière rigoureuse et précise.

Lorsque le résultat de la conception est exprimé dans un formalisme suffisamment précis, le système d'information opérationnel peut être dérivé quasi-automatiquement de ces spécifications.

Phase d'utilisation et de maintenance

Une fois le système en place et adapté à l'organisation, le travail ne s'arrête pas: il convient maintenant de faire évoluer le système en fonction des besoins de l'organisation, et de son évolution.

Cette phase mobilise plus ou moins d'énergie, tant de la part des informaticiens que de l'organisation, en fonction des phases précédentes. L'impact d'une bonne ou mauvaise conception se fait particulièrement ressentir à ce niveau.

0.2. Les méthodes de conception

Obtenir un système d'information opérationnel, correct et correspondant aux besoins de l'organisation, n'est pas chose aisée. En effet, les systèmes à construire sont devenus, au cours du temps, de plus en plus complexes et volumineux.

En plus de cette complexité, il est nécessaire de pouvoir faire évoluer les systèmes réalisés au même rythme que l'organisation représentée.

Si l'on veut conserver la fiabilité des systèmes d'information, il faut les concevoir de manière structurée, en suivant les étapes proposées par une méthode.

L'utilisation d'une méthode fournit un ensemble d'étapes, de concepts et de modèles, permettant une définition rigoureuse des actions à entreprendre tout au long du cycle de développement du logiciel.

Une méthode propose des modèles, un langage, une démarche, et éventuellement des outils .

Modèles et langage

Un modèle est un ensemble de concepts et de contraintes portant sur l'utilisation de ces concepts, destinés à bâtir une représentation des éléments du système d'information.

Le langage permet d'exprimer cette représentation de l'organisation dans un formalisme textuel précis (sans ambiguïté) et clair.

Le formalisme (graphique ou textuel) utilisé doit être aisément compris et sans ambiguïté, et les règles doivent permettre de définir une représentation dont les éléments sont sans contradiction entre eux.

Une Démarche

La démarche associée à une méthode propose des règles générales ou des étapes d'utilisation des modèles, du langage, et des éventuels outils.

L'objectif de la démarche est d'aboutir, par l'application des règles et des étapes proposées, à la réalisation de la phase de conception.

Des outils

Eventuellement, la méthode peut permettre l'utilisation d'outils informatiques pour faciliter les tâches de représentation graphiques et de validation.

0.2.1. Classification des méthodes

Bon nombre de méthodes ont été développées. On peut cependant les classer selon leur approche des systèmes d'information et leur démarche. La typologie présentée ci-dessous est celle proposée par C. ROLLAND dans [ROL. 88].

Méthode cartésienne

Les méthodes cartésiennes sont caractérisées par leur démarche et leur vision du système d'information.

La démarche est considérée comme une suite d'étapes à accomplir. Chaque étape est à son tour décomposée. La décomposition obtenue permettra de décrire divers aspects du système de manière globale (niveau de décomposition faible) ou plus précise (niveau de décomposition élevé).

On obtient donc une suite d'étapes élémentaires précisant les actions à accomplir et les documents à produire.

La vision du système généralement adoptée par les méthodes cartésiennes est une vision fonctionnelle. On voit le système d'information comme un ensemble de fonctions, de traitements à exécuter. Le système est donc un processeur de traitement de l'information.

Sur base de l'énoncé des fonctions, on appliquera la suite d'étapes élémentaires, on produira les documents intermédiaires (entre étapes) et finaux.

Les méthodes "cartésiennes" sont quelques fois qualifiées de "démarche fonctionnelle" ou "d'approche structurée". Les principales méthodes cartésiennes sont:

- SSA: Structured System Analysis de GANE & SARSON [GAN.79].
- SADT: Structured Analysis & Design Techniques.
- AXIAL: développée par IBM France- et IBM Belgique [PEL. 86].
- CORIG: dont MERISE, bien qu'étant une méthode systémique, est fortement inspirée. CORIG a été développée par Robert MAILLET (Compagnie Générale d'Informatique).

Méthode systémique

Les méthodes systémiques perçoivent le système d'information comme un modèle de la réalité organisationnelle. Le système d'information est abordé au travers d'une approche systémique (globale) de l'organisation et non plus seulement fonctionnelle.

Le système d'information est une représentation cohérente, complète, structurée et non redondante du système opérant de l'organisation, et est le support de son système de pilotage.

Les méthodes systémiques préconisent une représentation des phénomènes pertinents du système opérant de l'organisation, qu'ils soient, ou non, destinés à être informatisés. Cette représentation se fera via la réalisation de divers modèles.

La démarche qu'elles proposent est, en règle générale, moins stricte quant à l'ordonnancement des étapes de conception.

Les principales méthodes systémiques sont:

- MERISE
- IDA: Interactive Design Approach
- REMORA

0.3. Objectif du travail

L'objectif de notre travail est de réaliser un méta-modèle reprenant les concepts définis dans diverses méthodes de conception de système d'information. Ce méta-modèle étant destiné à servir de fondement à une interface entre outils de conception et outils de réalisation.

Ce travail se compose de trois parties:

Partie I. Présentation des méthodes

Les cinq méthodes que nous présenterons sont: MERISE, INTERACTIVE DESIGN APPROACH (IDA), REMORA, OBJECT ORIENTED ANALYSIS (OOA), et STRUCTURED ANALYSIS & DESIGN TECHNIQUES (SADT).

Ces méthodes font parties des méthodes les plus connues et les plus utilisées, en France. Pour chacune d'elles, nous présenterons les concepts et modèles utilisés lors de la phase de conception, ainsi que la démarche générale proposée par la méthode.

La présentation des différentes méthodes de conception des systèmes d'information nous permettra de mieux les comprendre, de mieux percevoir leurs points forts, leur spécificité.

Partie II. Elaboration d'un Méta-modèle global

Pour chaque méthode présentée, nous élaborerons un méta-modèle reprenant les concepts définis. Ces méta-modèles seront réalisés dans un formalisme entité/association [CHE. 76] tel que celui décrit par F. BODART et Y. PIGNEUR [BOD. 89].

Le méta-modèle réalisé pour chaque méthode, nous permettra d'avoir une vision compacte et précise du contenu conceptuel de cette méthode.

Nous intégrerons ensuite ces méta-modèles en un méta-modèle global. Pour ce faire, nous choisirons un méta-modèle parmi ceux réalisés pour les méthodes, et nous lui adjoindrons les concepts nouveaux apportés par les autres méthodes.

L'élaboration de ce méta-modèle global va permettre de visualiser les ressemblances entre méthodes. On verra ainsi quels sont les concepts fondamentaux utilisés pour la modélisation des informations et des traitements.

De plus, ce méta-modèle global nous permettra de mettre en relation des concepts semblables mais souvent présentés de manière différente.

Par rapport à d'autres travaux semblables [ROC. 89], nous nous distinguerons par une étude incluant une méthode récente (Object Oriented Analysis), ainsi que par une présentation limitée aux aspects conceptuels issus de méthode et non pas d'outils CASE (Computer Aided Software Engineering).

Partie III. Conception d'outils

Enfin, nous présenterons le programme LCI (LINC CASE INTERFACE), développé par la société UNISYS. LCI constitue une interface entre les outils de conception existant sous forme de CASE, et le langage de quatrième génération LINC.

Ce programme préfigure l'évolution future des outils et des méthodes de conception de systèmes d'information.

Le méta-modèle élaboré pourrait constituer une base pour l'élaboration du "dictionnaire" intermédiaire de programmes semblables à LCI.

Par rapport au "dictionnaire" présent dans le programme LCI 1.0, notre méta-modèle est une extension. Nous nous sommes basé sur les concepts présents dans diverses méthodes de conception, et non pas uniquement sur ceux présents dans les outils CASE, et de plus, nous avons également abordé les aspects de représentation du comportement dynamique du système.

Partie I

Présentation des méthodes

1. Merise

1.1. Introduction

A la fin des années 70, le ministère de l'Industrie français lance une consultation afin de concevoir une méthode de développement "d'intérêt national". La méthode MERISE est définie et mise au point durant les années '78 et '79.

L'utilisation et la diffusion de MERISE dans les administrations et les entreprises publiques, encouragées par la Mission à l'Informatique, ont contribué à faire de cette méthode un standard de fait en France.

MERISE ne propose pas de décomposition hiérarchique des traitements. La future version de MERISE, MERISE II, inclura la notion d'approche par raffinements successifs des traitements.

L'exposé qui suit est basé sur le livre de H. TARDIEU, A. ROCHFELD et R. COLLETTI [TAR. 89], ainsi que, dans une moindre mesure, sur celui de Y. TABOURIER [TAB. 86].

1.2. Concepts et modèles

1.2.1. Concepts "statiques"

La représentation des aspects statiques du système d'information va se faire grâce à l'utilisation du "formalisme individuel" apparenté au formalisme entité-relation décrit par CHEN [CHE. 76].

Les concepts de base sont les individus, les propriétés et les relations: [Tar. 86 pp. 135 et 136]

Les individus

"Un individu (ou entité) est un objet abstrait ou concret dans l'univers du discours" [TAR. 89]. A partir de cette définition, on va définir les classes d'individus ou les "individus-types".

"Un individu-type est une classe d'individus qui ont en commun un ensemble de propriétés". Un individu ne peut appartenir qu'à un seul individu-type sauf dans le cas d'une spécialisation (Cf. infra).

Les relations

"Une relation est une association perçue entre individus dans l'univers du discours". "Une relation-type est une relation définie sur un ou plusieurs individus-types".

La dimension d'une relation-type est le nombre d'occurrences d'individus concernés par une occurrence de la relation-type. Une relation-type de dimension 2 est dite "binaire", de dimension 3 "ternaire", etc...

A partir des notions de fonctionnalité et de partialité/totalité, on associe à une relation-type ses cardinalités maximum et minimum.

La cardinalité minimum (maximum) d'une relation-type est le nombre minimum (maximum) de fois où chaque occurrence d'un individu-type participe (peut participer) à une occurrence de cette relation-type.

Les propriétés

"Une propriété est un attribut que l'on perçoit sur un individu ou sur une association entre individus dans l'univers du discours (...) Les propriétés sont spécifiques à un individu, et une propriété peut avoir une valeur".

"Une propriété-type est une classification des propriétés semblables de toutes les occurrences d'individus appartenant à un individu-type ou de toutes les occurrences de relations appartenant à une relation-type".

Parmi les propriétés que possède un individu-type, il en est une qui permet de distinguer (d'identifier) de manière unique toute occurrence de cet individu-type. Il s'agit de l'identifiant.

Dépendance fonctionnelle

"La dépendance fonctionnelle inter-individus est un cas particulier de relation binaire. Elle traduit le fait que connaissant une occurrence (*source*) de l'un des deux individus composant la collection de la relation, on connaît directement une et une seule occurrence de l'autre individu (*cible*)".

Les cardinalités sont (0,1) ou (1,1) pour les individus sources et quelconque pour les individus cibles. Dans le cas où la cardinalité minimale des individus sources est 1, respectivement 0, on parle de dépendance fonctionnelle forte, respectivement faible.

Une contrainte d'intégrité est dite stable si une fois le lien établi entre deux individus, ce lien ne peut être modifié dans le temps.

Lorsqu'une dépendance fonctionnelle est forte et stable, on parle de *Contrainte d'Intégrité Fonctionnelle* inter-individus.

Généralisation et spécialisation

Le concept de "généralisation" permet "d'appréhender une classe d'objets individuels comme générique", c'est-à-dire de représenter les notions de sur- et sous-type. Par ce mécanisme, on exprime que toutes les occurrences d'un individu-type appartiennent à un ensemble d'occurrences rattaché à un autre individu-type, dit générique.

Tout individu-type hérite des propriétés et relations définies sur son "individu-type générique".

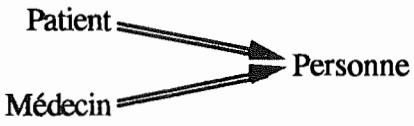


Figure 1.1. Généralisation & héritage.

Dans la figure 1.1 ci-dessus, les patients et les médecins sont des personnes. Cela signifie que les individus-types Patient et Médecin héritent des propriétés et relations associées à l'individu-type Personne.

Le concept de spécialisation permet de spécifier qu'un individu appartient à plusieurs individus-types génériques. L'individu spécialisé hérite des propriétés des individus-types génériques.

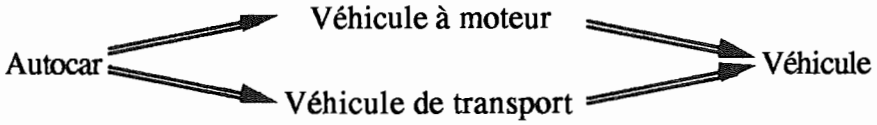


Figure 1.2. Généralisation & héritage.

Ainsi dans la figure 1.2 ci-dessus, l'individu *Autocar* hérite des propriétés et relations des individus *Véhicule de transport* et *Véhicule à moteur*.

On remarque que ces deux derniers individus-types doivent appartenir au même individu-type générique.

1.2.2. Concepts "dynamiques"

La description de la dynamique va se faire grâce à trois concepts: les événements, les opérations, et les synchronisations.

Les événements

Un événement indique au système d'information que quelque chose s'est passé, et que ce dernier doit réagir. L'événement "a lieu" lorsque le système d'information en prend connaissance.

Par exemple, un bon de commande n'existe que lorsqu'il est encodé, et non pas dès qu'il est rédigé par le client. Il peut donc y avoir un décalage entre le phénomène réel et sa perception par le système.

On distingue deux types d'événements en fonction de leur origine:

- les événements externes qui proviennent de l'environnement ou de l'univers du discours. Leur compte rendu est un message.
- les événements internes survenant lors de la terminaison d'une opération. Ces événements ne sont pris en compte que s'il est nécessaire de déclencher une nouvelle réaction en fin d'opération ou d'émettre un message à destination de l'environnement.

Chaque événement est une occurrence d'un événement-type. On associe à tout événement-type

- le type de commande qui détermine la ou les actions qui doivent être entreprises en réaction à cet événement
- le message-type associé à l'événement.

"Deux événements appartenant à un même événement-type seront distingués soit par les valeurs prises par les propriétés du message, soit, si ces valeurs sont identiques par le moment précis où l'événement s'est produit".

On associe également à un événement-type deux caractéristiques plus générales qui sont le nombre maximum d'occurrences simultanées de ce type d'événement, et la fréquence d'apparition des occurrences de ce type d'événement.

Les opérations

"Une opération est une action ou un ensemble d'actions accomplies par le processeur d'information en réaction à un événement".

Le cycle de vie d'une opération se déroule comme suit: l'opération est non-existante, survient un événement qui la déclenche. L'opération démarre et devient active, elle effectue ses actions, s'achève (fin d'opération), et passe dans l'état "terminée".

Une fois déclenchée, une opération est ininterruptible et ne doit attendre aucun autre événement.

Les actions composantes des opérations sont décomposées en actions élémentaires (Insertion, recherche, effacement) dont les paramètres seront les informations contenues dans les messages. Une fois terminée, une opération peut entraîner l'émission de un ou plusieurs événements.

Toute opération appartient à un type d'opération. Ce type spécifie notamment:

- les types d'événements internes et/ou externes à émettre en fin d'exécution ainsi que les règles (conditions) d'émission par l'opération de ces événements.
- l'action à exécuter. Cette action sera décrite sur base des actions élémentaires (Insertion, effacement, recherche), et pourra faire appel aux structures élémentaires de la programmation structurée (SI - ALORS - SINON ; TANT QUE - ; ...). L'action effectivement réalisée dépendra des valeurs des propriétés contenues dans le message.
- La durée d'exécution de l'opération.

Les synchronisations

Les synchronisations représentent le fait que plusieurs événements doivent être présents avant de démarrer une opération. On leur associe une règle de réalisation. Cette règle exprime la ou les conditions que les informations contenues dans les messages associés aux événements doivent vérifier pour pouvoir déclencher l'opération associée à la synchronisation.

"L'arrivée du premier événement contributif met la synchronisation en attente. Cette attente se poursuit jusqu'à l'arrivée du dernier événement. Si la ou les règles sont vérifiées, la synchronisation est activable. Elle passera en mode activée de manière synchrone ou asynchrone[1]. Une fois activée, la synchronisation déclenche l'opération et attend l'arrivée d'un événement contributif pour passer à nouveau en attente.

Chaque synchronisation appartient à une synchronisation-type. Pour cette dernière, on détermine:

- Les types d'événements internes et/ou externes contribuant à la réalisation de la synchronisation.
- La ou les conditions de réalisation sous forme de proposition logique.

---

[1] Le mode synchrone implique que le moniteur dynamique vérifie à intervalle régulier la règle associée à la synchronisation et active cette dernière lorsque la règle est vérifiée. Le mode asynchrone implique que l'arrivée du dernier événement active directement la synchronisation.

- Les conditions locales c'est-à-dire le critère de sélection[2] de l'occurrence qui sera prise en considération pour vérifier la condition de réalisation de la synchronisation parmi toutes les occurrences présentes d'un événement-type.
- La durée limite de participation du premier événement contribuant à la synchronisation.
- Le délai de synchronisation imposé entre le moment où la synchronisation devient activable et le moment où elle est activée.

1.2.3. Les modèles

Dans MERISE, on définit deux modèles dynamiques (modèles des traitements) et deux modèle statiques (modèles de données).

Modèle Conceptuel des Traitements

Pour réaliser le modèle conceptuel des traitements (MCT), on élabore une matrice des flux circulant entre les différents acteurs réels concernés par le système d'informations. Le contenu de cette matrice, représenté graphiquement est appelé le *modèle conceptuel de communication* [3].

On ordonne ensuite les flux en identifiant les flux directement engendrés par les acteurs (flux primaires) et les flux secondaires (flux découlant des primaires). On obtient ainsi un graphe orienté, dont les sommets sont les événements du modèle dynamique des traitements et les arcs les futures opérations-types.

En spécifiant les opérations et les synchronisations associées, et en remplaçant les arcs par les représentations des opérations, des synchronisations, et des flux de données, on obtient le modèle conceptuel des traitements.

Graphiquement, l'opération est représentée par un rectangle dont la première ligne contient son nom. La partie basse du rectangle peut être subdivisée en zones correspondant aux conditions d'émissions des résultats.

Les événements sont représentés par un cercle contenant leur nom, et éventuellement un nombre indiquant la capacité du type d'événement.

Les synchronisations sont représentées graphiquement par un symbole en forme d'entonnoir dans lequel "arrivent" les messages (Figure 1.3.b).

Les flèches décrivent les flux d'informations dans le modèle.

Dans l"exemple ci-dessous (Figure 1.3.a), lorsque l'opération "*Traitement* " se termine et que la condition *Condition_1* est vérifiée, alors l'événement E_1 est généré

---

[2] Ce critère se base uniquement sur les valeurs contenues dans le message associé aux événements-types.

[3] TABOURIER Y., op. cit.

par l'opération. Il en est de même pour *Condition_2* et E_2 . Il faut remarquer que les conditions d'émission des événements peuvent être plus nombreuses que dans l'exemple ci-dessous. De plus, elles peuvent ne pas être mutuellement exclusives.

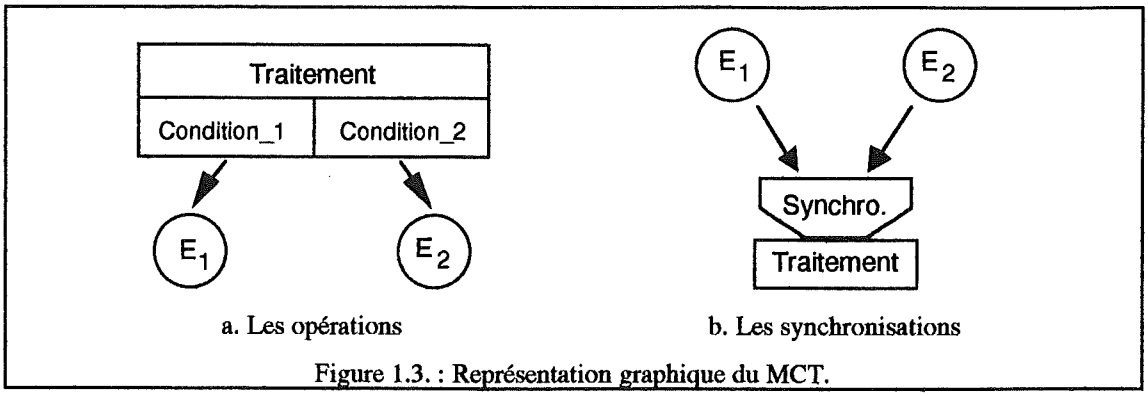


Figure 1.3. : Représentation graphique du MCT.

On peut également représenter les accès à la base de données par des flèches, et la base elle-même par un cylindre. On peut associer, dans ce cas, un nom significatif à l'accès ainsi spécifié.

Modèle Organisationnel des Traitements (MOT)

Ce modèle est destiné à représenter la nouvelle organisation des tâches engendrée par le nouveau système d'information mis en place dans l'entreprise analysée.

Le MOT contient la description des procédures de traitement, c'est-à-dire la façon dont les opérations s'enchaîneront réellement. Pour ce faire, on va décrire des processus qui sont des ensembles d'opérations conceptuelles vues comme un tout à organiser. Plusieurs variantes (plusieurs "procédures organisées") seront peut être possibles pour décrire un même processus.

La description d'une opération "organisée" comprendra, en plus des éléments de sa description conceptuelle:

- la cellule de l'entreprise chargée de son exécution (localisation, moment d'exécution),
- le type de moyens techniques requis par l'opération (machine, ordinateur),
- le type de moyens humains (éventuellement profil de l'opérateur).

Dans ce modèle, les synchronisations restantes représentent les délais entre opérations (deux mois après la facturation, trois jours après la livraison, tous les jours à 17h00, ...). L'enchaînement de ces éléments est représenté par des flèches.

Il faut remarquer que certaines opérations conceptuelles peuvent être éclatées ou regroupées au niveau organisationnel.

Par exemple l'enregistrement des bons de commandes peut constituer une opération conceptuelle unique, mais donner naissance à des procédures organisationnelles différentes selon qu'il s'agit d'une commande téléphonique ou écrite.

Inversement, diverses opérations conceptuellement différentes peuvent être regroupées en une seule opération organisationnelle.

Il est à noter que le modèle organisationnel permet de retrouver les traitements qui seront à implémenter, ainsi qu'une description de ces traitements dont la précision et le degré d'analyse dépendent uniquement du concepteur.

Modèle Conceptuel des Données

Le formalisme graphique utilisé est celui des modèles entités-relation. Un individu-type est représenté par un rectangle contenant son nom dans sa partie supérieure, son identifiant et la liste de ses propriétés dans la partie inférieure (Figure 1.4). Sur la même figure , les individus-type *Patient* et *Médecin* sont des sous-types de *Personne* .

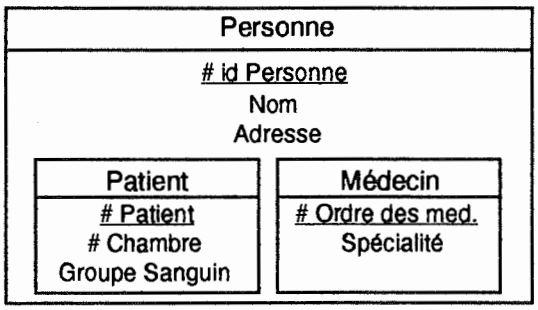


Figure 1.4. Représentation des individus-type et des sous-types

Une relation-type est représentée par un ovale (ou un hexagone selon le formalisme graphique) contenant son nom (Figure 1.5). Les cardinalités sont inscrites sur les arcs reliant l'individu à la relation et se lise du côté de l'individu. Ainsi sur la figure 1.5 ci dessous, tout client commande au moins un produit, et un produit peut être commandé par zéro, un ou plusieurs clients.

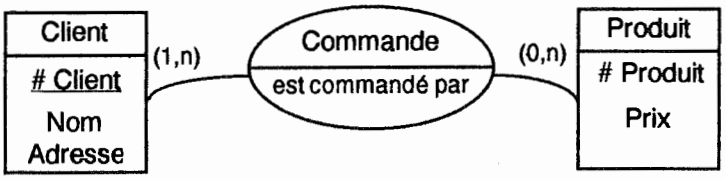


Figure 1.5. Représentation graphique des relations.

Remarque:

- Dans le cas des relations binaires sans propriété, on inscrit le nom de la relation sous forme passive dans la partie inférieure de l'ovale.

Modèle Logique des Données

"Le modèle logique est une représentation du modèle conceptuel en fonction des possibilités techniques du moment"[4].

On va donc transformer le modèle conceptuel en un modèle relationnel, les individus devenant des relations aux sens relationnel, leur identifiant y devenant clé primaire. Les relations du modèle conceptuel vont devenir des relations (au sens relationnel), leur clé primaire étant la concaténation des identifiants des individus que le relation conceptuelle joignait.

1.3. Démarche

1.3.1. Les cycles

La méthode Merise identifie trois cycles: le cycle de vie, le cycle d'abstraction et le cycle de décision.

Le cycle de vie

Le cycle de vie définit le passage du système réel au système "artificiel". La conception du système d'information artificiel définit ce passage, et le système artificiel, une fois mis en œuvre peut modifier le système réel.

Ce cycle est décomposé en trois étapes:
- la conception du S.I. pour obtenir une description fonctionnelle et technique
- la réalisation qui concerne l'élaboration des programmes et des procédures à mette en œuvre
- la maintenance du système d'informations

Le cycle d'abstraction

Ce cycle permet "d'isoler, à un niveau spécifique, les éléments significatifs contribuant à la description d'un système cohérent, tout en ignorant les détails".

Trois niveaux pour les données et trois niveaux pour les traitements sont mis en évidence dans MERISE.Pour les données, ce sont les niveaux conceptuel, logique et physique; pour les traitements, les niveaux conceptuel, organisationnel, et opérationnel. Chaque niveau venant détailler le précédent en y ajoutant certains aspects volontairement ignorés jusque là.

---

[4] TARDIEU H., A. ROCHFELD A., COLLETTI R., op. cit.

Le cycle de décision

Il rend compte des choix effectués lors du cycle de vie du système. Ce cycle permet le pilotage du projet par les développeurs. Ce pilotage est basé sur diverses évaluations: coût du projet, durée de développement, solutions techniques possibles, impact de la mise en œuvre du système, ...

De plus, on hiérarchise les décisions afin de satisfaire les objectifs établis.

1.3.2. Etapes de développement

Schéma directeur

Le but de cette première étape est d'établir un lien entre les objectifs de l'organisation et ses besoins en information. On définit les domaines du système d'information, leur architecture, un plan de développement de chaque modèle, ainsi qu'un calendrier des programmes à développer.

Etude préalable

Cette étude consiste en une étude de faisabilité et d'utilité fonctionnelle du système d'information. Elle vise à déterminer si le système d'information souhaité est réalisable et s'il réalisera les objectifs qui lui sont assignés.

Etude détaillée

Le but est d'obtenir les spécifications fonctionnelles détaillées du système. Lors de cette étape, on va élaborer les modèles conceptuels des données (MCD) et des traitements (MCT). De plus, on va décrire les écrans et les rapports imprimés.

Etude technique & Production du logiciel

L'objectif de cette phase est de réaliser les programmes, et de définir l'organisation physique des fichiers de données. Pour ce faire, on va construire les modèles logique et physique des données, et les modèles organisationnel et opérationnel des traitements.

Mise en œuvre

Cette phase reprend tous les éléments à élaborer pour une mise en œuvre efficace, avec notamment la création des fichiers de base, la formation des utilisateurs, les

étapes de transition entre les systèmes, la planification des modifications d'organisation du travail, ...

Maintenance

Il s'agit maintenant de faire évoluer le système installé en se basant sur les impacts du nouveau système, les demandes des utilisateurs.

---

2. Interactive Design Approach

2.1. Introduction

La méthode IDA™ (Interactive Design Approach) est le fruit de travaux menés, depuis 1977, conjointement par F. BODART et Y. PIGNEUR [BOD. 89].

La méthode IDA prend en considération les étapes du début du cycle de vie d'un système, à savoir les étapes qui recouvrent l'analyse fonctionnelle (analyse d'opportunité et analyse conceptuelle). Les étapes suivantes, c'est-à-dire de réalisation, consistent en la "déduction" de solutions à partir des résultats de l'analyse fonctionnelle, et ne présentent plus le caractère créatif des deux premières étapes.

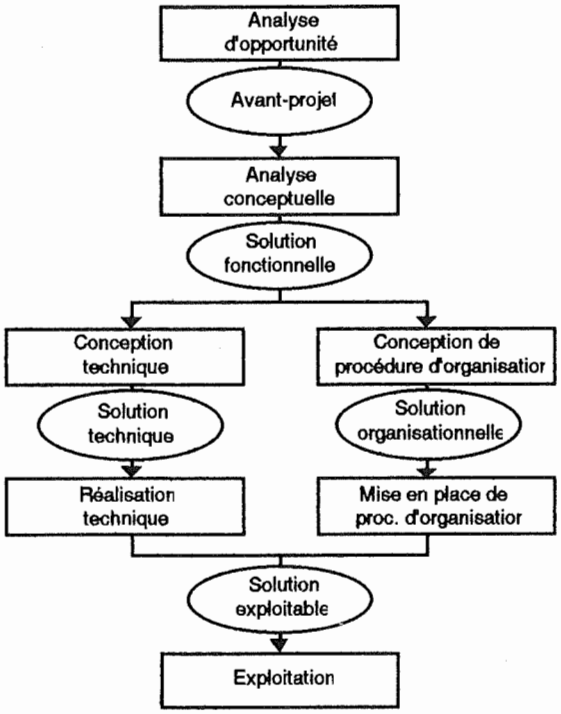


Figure 2.1. Etapes du cycle de vie d'un S.I. (IDA). Extrait de [BOD. 89] p.5.

La méthode IDA offre un langage de spécification nommé DSL (Dynamic Specification Language). Ce langage permet une description graphique et textuelle des différents éléments représentés. La description textuelle permet de compléter le graphique par des commentaires ou des contraintes d'intégrité difficilement représentables graphiquement.

Enfin, la description textuelle permet de faciliter l'acquisition des descriptions réalisées par les outils logiciels associés à la méthode (DSL/SPEC).

2.2. Concepts et modèles

Six modèles sont proposés afin de représenter tous les aspects du système d'information.

2.2.1. Modèle Entité-Association

Ce premier modèle est destiné à définir la sémantique des données mémorisables et véhiculables du système d'information.

Les concepts utilisés dans ce modèle sont ceux d'entité, d'association, et d'attribut.

Concept d'entité

Entité: Une entité est une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité n'existe en tant que telle que par rapport à un individu ou un groupe qui la considère comme un tout, lui confère une existence autonome et la distingue d'autres entités et de son environnement. Une entité peut posséder des attributs, et est caractérisée par une durée de vie.

Type d'entité: Classe de toutes les entités possibles du réel perçu qui vérifient la définition constitutive du type. Les types d'entité ne forment pas nécessairement des classes disjointes.

Ensemble d'entité: classe des entités du même type qui existent à un instant particulier.

Concept d'association

Association: une association est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné.

L'association est dépendante de l'existence des entités qu'elle relie.

Type d'association: classe de toute les associations possibles du réel perçu qui vérifient la définition constitutive du type.

Le degré du type d'association est le nombre de types d'entités, non nécessairement distincts, sur lesquels le type d'association est défini.

Concept d'attribut

Attribut: caractéristique ou qualité d'une entité ou d'une association. Il peut prendre un(e) ou plusieurs valeurs ou groupes de valeurs.

Valeur: symbole utilisé pour représenter un fait élémentaire; il prend souvent la forme d'une chaîne de caractères.

Type de valeur: classe ou domaine de toutes les valeurs possibles qui vérifient la définition constitutive du type; cette définition peut être donnée soit par la liste des éléments de la classe (définition en extension), soit par la propriété à laquelle les membres de la classe doivent satisfaire (définition en compréhension).

Un ensemble de valeur est la classe des valeurs de même type qui existent à un instant particulier.

Un attribut possèdent divers propriétés:

- il peut être simple ou répétitif: un attribut est simple si pour une entité ou une association il ne peut prendre qu'une seule valeur. S'il peut prendre plusieurs valeurs d'un même type, alors il est répétitif.
- Attribut élémentaire ou décomposable: un attribut est décomposable si à une entité ou une association, il fait correspondre un groupe de valeurs de type différents et peut être décomposé, au plus, en autant d'attributs qu'il y a de types différents dans le groupe de valeurs. Un attribut non-décomposable est élémentaire.
- Attribut obligatoire ou facultatif: un attribut est facultatif s'il peut ne pas prendre de valeur pour certaines occurrences du type qu'il caractérise, c'est-à-dire que cet attribut n'a pas de signification pour ces occurrences. Si tel est le cas, on donnera à pareil attribut une valeur "inexistante". (A ne pas confondre avec la valeur "inconnue").

Contraintes d'intégrité

Une contrainte d'intégrité est une propriété, non représentée par les concepts de base du modèle que doivent satisfaire les données appartenant à la mémoire du système d'information.

Ces contraintes peuvent être statiques, c'est-à-dire vérifiée à tout instant, indépendamment des changement intervenant dans la base de données, ou dynamique. Dans ce cas, elles définissent la validité des changements d'état de la base de données.

On rencontre deux contraintes d'intégrité obligatoires: la connectivité et l'identification. Les autres contraintes d'intégrité sont facultatives.

Connectivité

Soit $R(ro_1:E_1, \ldots, ro_i:E_i, \ldots, ro_n:E_n)$ un type d'association, défini sur les types d'entités $E_1, \ldots, E_i, \ldots E_n$ où chaque E_i joue un rôle ro_i.

La connectivité de R est définie par un ensemble de couples d'entiers (min_i,max_i), $1 \leq i \leq n$, où

• min_i indique le nombre minimum de fois que, à tout moment, toute occurrence de E_i doit assumer le rôle ro_i; c'est-à-dire le nombre minimum d'occurrences de R auquel toute occurrence de E_i doit participer. ($min_i \geq 0$)

• max_i indique le nombre maximum de fois que, à tout moment, toute occurrence de E_i peut assumer le rôle ro_i; c'est-à-dire le nombre maximum d'occurrences possible de R pour toute occurrence de E_i. ($max_i \geq 1$, $max_i \geq min_i$)

Identification

L'identifiant d'un type d'entité ou d'un type d'association permet d'identifier de manière univoque chaque occurrence de ce type.

Pour un type d'entité, l'identifiant peut être un attribut, un groupe d'attributs, un type d'association auquel participe le type d'entité (La voiture conduite par M. Dupont le 10 mai 1991 à 17h34), ou un rôle joué dans une association et un ou plusieurs attributs. Il peut également être identifié par un attribut substitut, c'est-à-dire ne correspondant à aucun fait de l'organisation (exemple: *numéro de patient*).

Un type d'association est identifié par l'identification des types entités sur lesquels il est défini.

Existence

On peut lier la validité de l'existence d'une occurrence d'un type d'entité ou d'un type d'association à des éléments de la base de données du système autres que la définition de ce type.

Inclusion

Cette contrainte permet de représenter le fait que si une entité joue un rôle déterminé dans une association, alors cette entité doit également jouer un rôle précis dans une autre association.

Par exemple, dans la figure 2.2 ci-dessous, une *Personne* qui loue une *Voiture* doit posséder un *Permis de conduire*.

Exclusion

Cette contrainte permet de modéliser le fait que différents rôles, qui pourraient être joués par les mêmes occurrences d'un type d'entité, sont mutuellement exclusifs.

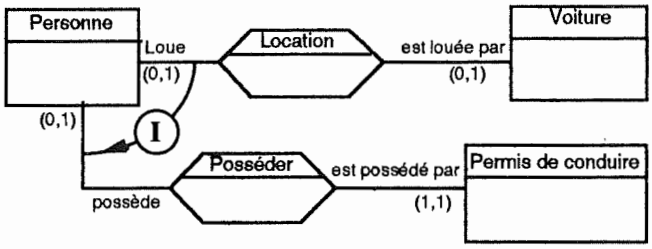


Figure 2.2. Exemple d'Inclusion dans IDA.

Egalité de rôles

On modélise ainsi le fait que si une entité joue un rôle donné, alors elle doit également jouer un autre rôle, et inversement. On a, en fait, une inclusion dans les deux sens.

Ces trois contraintes (inclusion, exclusion et égalité) peuvent également porter sur des associations.

Sous-type

Si E_1 un type d'entité est un sous-type de E_2, alors toute occurrence de E_1 est aussi une occurrence de E_2, et cette occurrence hérite des propriétés de E_2. On le modélise par un type d'association prédéfini entre E_1 et E_2, où E_1 joue le rôle "est sous-type de...", et E_2 le rôle "a pour sous-type...". La connectivité de cette association est (1,1) pour le rôle joué par le sous-type, et (0,1) pour le rôle joué par le sur-type.

Domaine de valeurs

Ces contraintes réduisent l'ensemble des valeurs que peut prendre un attribut d'un type d'entité, ou d'un type d'association.

Dépendance fonctionnelle entre attributs et entre rôles

L'attribut A_1 du type d'entité E (ou d'association) dépend fonctionnellement de l'attribut A_2 de E, si à chaque valeur de A_2 correspond une seule valeur de A_1, et ce à tout moment. L'attribut A_2 est appelé le déterminant et A_1 le déterminé. On peut utiliser un groupe d'attributs en tant que déterminant.

L'identifiant d'un type d'entité est déterminant pour chaque attribut.

Les dépendances fonctionnelles peuvent aussi être définies sur les rôles de type d'association, ou être définies entre des attributs et des rôles: dépendance mixte.

De plus, on parlera de dépendance fonctionnelle totale, et de dépendance multivaluée. Il y a dépendance fonctionnelle totale lorsque la dépendance est telle que si l'on retire un élément du groupe des déterminants alors cette dépendance cesse d'exister. On aura une dépendance multivaluée quand le déterminé est un attribut répétitif, ou un rôle multivalué

Modèle Entité-Association

Ce modèle reprend les concepts décrits ci-dessus, dans un schéma graphique, éventuellement accompagné de précisions sous forme textuelle. Ce sera notamment le cas lorsque les contraintes à représenter sont trop complexes pour être décrites de manière claire et complète sur le graphique.

Un type d'entité est représenté par un rectangle contenant dans sa partie supérieure le nom du type d'entité, et dans sa partie inférieure le nom des attributs. Les identifiants seront, en cas d'ambiguïté soulignés.

Un type d'association est représenté par un hexagone. Cet hexagone est relié aux rectangles représentant les types d'entités sur lesquels est défini le type d'association. Sur la patte qui relie type d'entité et type d'association sont indiqués le rôle joué par le type d'entité, ainsi que la connectivité associée.

La contrainte d'inclusion est représentée par une flèche joignant les deux rôles, sur laquelle figure un cercle contenant la lettre "I" (Cf. figure 2.2 ci-dessus).

a. Exclusion b. Egalité

Figure 2.3. Contrainte d'exclusion et d'égalité dans IDA.

Pour la contrainte d'exclusion, le cercle contient un dièse (Figure 2.3.a.), et il contient le signe "=" pour une contrainte d'égalité (Figure 2.3.b.).

Les sous-types sont représentés comme montré dans l'exemple ci-dessous, où *Patient* et *Médecin* sont des sous-types de *Personne* (figure 2.4). On remarque que dans notre exemple, un *Médecin* ne peut pas être *Patient*.

Les dépendances fonctionnelles sont représentées par des flèches de l'attribut ou du rôle déterminant vers le déterminé. On associe à ces flèches le nom du type d'entité ou d'association pour lequel cette dépendance à lieu.

Les dépendances multivaluées sont représentées par des flèches à double pointe.

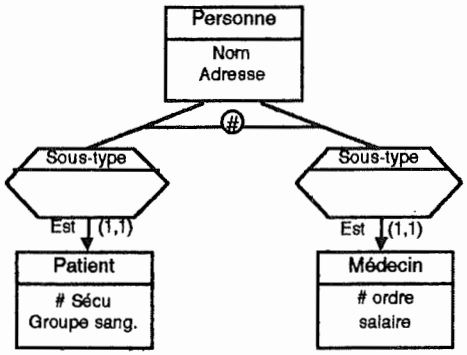


Figure 2.4. Sous-typage dans IDA.

2.2.2. Modèle de structuration des traitements

La méthode IDA, tout comme "STRUCTURED ANALYSIS" et SADT, permet une structuration hiérarchique des traitements. Cependant, IDA propose des niveaux prédéfinis: Projet, Applications, Phases et Fonctions.

L'objectif de ce modèle est de fournir une représentation structurée des traitements, dans laquelle chaque traitement est décomposé sous forme arborescente (tout traitement provient de la décomposition d'un seul traitement de niveau supérieur, et est décomposé en au moins un traitement de niveau inférieur).

Chacun de ces traitements "comprend un ensemble de règles à suivre ou d'actions à entreprendre pour réaliser une fonctionnalité du système d'information". On prendra soin de ne pas introduire de contradiction entre les traitements de niveau différents mais hiérarchiquement apparentés.

Pour chaque traitement, on va donner son nom, son objectif, ses performances (attendues), et son niveau hiérarchique.

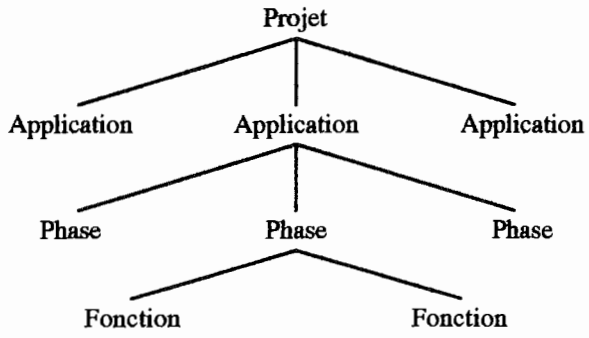


Figure 2.5. Hiérarchie des traitements dans IDA.

• Le niveau "projet"

"Un projet est la partie du système qui fait l'objet d'une analyse. C'est un sous-système du S.I. à réaliser".

• Le niveau "application"

Une application est un traitement quasi-autonome par rapport aux autres applications d'un projet: elle constitue une unité de planning dans le gestion d'un projet. Une application constitue la plus petite partie d'un projet dont le cycle de développement doit être considéré globalement".

Deux critères sont donnés pour identifier les applications:

- Faible interaction avec d'autres applications: communications ponctuelles, par échange d'information; mais circulation d'information intense au sein-même de l'application.
- Liée à "un flux homogène de message et/ou à une structure d'information homogène".

• Le niveau "phase"

La phase est le concept central de cette nomenclature des traitements. "Une phase est un traitement possédant une unité spatio-temporelle d'exécution".

Une phase doit être entièrement exécutée dans une cellule d'activités[5].

On dispose de deux critères d'identification:

- Unité spatiale d'exécution: pas de changement spatial ni de ressource lors de l'exécution.
- Unité temporelle d'exécution: exécution (théoriquement) non-interruptible. Cela implique qu'il ne peut y avoir lors de l'exécution d'une phase de point d'attente, ni de différence de périodicité.

• Le niveau fonction

Une fonction correspond au niveau élémentaire dans cette nomenclature. On lui associe un objectif et un comportement.

Une fonction correspond selon l'acteur du système d'information à:

- pour l'organisateur: un objectif de gestion élémentaire,

---

5 "Centre d'activité homogène dans le temps et dans l'espace, doté de ressources humaines et/ou matérielles et pourvu de règles de comportement nécessaires à son fonctionnement".

- pour l'utilisateur: un échange indécomposable de messages entre lui et le système d'information,
- pour le concepteur: une action minimale portant sur les données mémorisées.

On précisera les messages (et leurs propriétés) reçus en entrée, et produits en sortie pour chaque fonction.

Les critères d'identification des fonctions sont:

- Agrégabilité: chaque fonction doit correspondre à un sous-objectif de la phase, et l'ensemble des fonctions, composantes d'une phase, doit couvrir les objectifs associé à cette phase.
- Ergonomie: chaque fonction pourra être vue comme un service par l'utilisateur. Ce service demandera, pour son bon déroulement, le respect de certaines règles, de la part de l'utilisateur, en ce qui concerne les informations à fournir. La fonction se déroule sans interaction, et produira ses résultats en fin d'exécution.
- Cohérence: toute fonction doit, au terme de son exécution, laisser la mémoire du système d'information dans un état cohérent.

2.2.3. Modèle de la statique des traitements

Dans ce modèle, les traitements sont d'abords vus comme des boites noires qui à partir d'inputs créent des outputs. On va donc, pour chaque traitement, énoncer sa définition, ses objectifs, les informations reçues en entrées et produites en sorties (messages et/ou mémoire du S.I.), et les actions primitives (réception d'un message, consultation et/ou mise à jour de la mémoire du S.I., génération d'un message).

Ensuite, on déterminera les règles de traitement pour chaque traitement (généralement de niveau fonction). On décrira les contrôles éventuels à effectuer sur les informations reçues, et les règles à appliquer afin de réaliser les objectifs du traitement décrit. Ces règles de traitement seront spécifiées de façon déclaratives plutôt qu'algorithmique.

Pour ce modèle, il n'y a pas de représentation graphique prédéfinie.

2.2.4. Modèle de la dynamique des traitements

Ce modèle représente les conditions d'exécution et l'enchaînement des processus.

Contrairement à MERISE, on distingue dans IDA les traitements de leur exécution (processus).

Processus et événements

"Un processus représente l'exécution d'une procédure correspondant à un traitement effectué dans les système d'information". Un type de processus regroupe tous les processus correspondant à un même traitement. Chaque processus est déclenché par un événement (génération d'un message ou changement d'état d'un processus: déclenchement ou terminaison), et cause des événements en sortie. Ces événements déclenchent éventuellement d'autres processus.

Pour chaque type de processus, est précisée la liste des types d'événements déclencheurs (classe d'événements demandant le même type de réaction du système). Un événement est un changement d'état du système porté à la connaissance d'un observateur par l'intermédiaire d'un message.

Les enchaînements

La distinction est faite entre trois types d'enchaînement : les enchaînements séquentiels, parallèles et convergents. L'enchaînement séquentiel correspond au déclenchement d'un processus par un processus qui se termine. L'enchaînement parallèle est le déclenchement de plusieurs processus simultanément, et un enchaînement est dit convergent si un processus peut être déclenché par plusieurs autres.

Synchronisation, sélection et duplication.

La synchronisation représente le fait qu'il faille attendre la survenance d'une combinaison d'événements avant de déclencher un processus (cette notion existe dans MERISE). IDA ajoute aux synchronisations les clauses de qualification (les événements déclencheurs doivent partager une propriété commune ; par exemple : même événement déclencheur initial), et celles de modalité de mémorisation (durée de contribution d'un événement à la réalisation d'une synchronisation).

La sélection représente le fait qu'un événement ne déclenche un traitement que si une condition donnée est satisfaite.

IDA permet également de représenter le fait qu'un événement déclenche plusieurs processus issus du même traitement en même temps, et ceci, grâce à la notion de duplication. Par exemple, la terminaison d'un approvisionnement déclenchera, pour chaque commande différée, sa réalisation.

Représentation graphique

Un type de processus est représenté par un rectangle contenant son nom (souvent celui du traitement correspondant — figure 2.6.a) .

Un message est représenté comme indiqué figure 2.6.b. Un message externe est tracé sur la ligne pointillée symbolisant la frontière entre le système et le monde extérieur.

Un événement est représenté par un point posé sur la bordure inférieure du message ou du processus dont il constate respectivement la génération ou la terminaison.

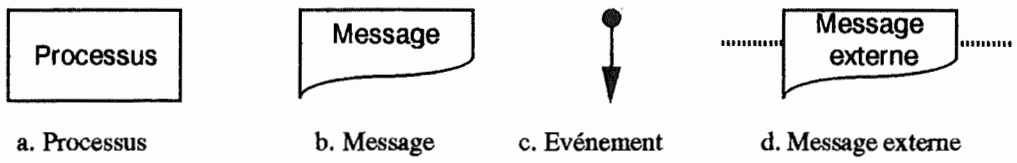


Figure 2.6. Processus, message et événements (IDA).

Dans la figure 2.7 ci-dessous, sont représentés divers type d'enchaînements. La figure 2.7.a montre un enchaînement séquentiel : la terminaison du processus *A* déclenche le processus *B*. Dans la figure b, le processus *A* déclenche les processus *B* et *C* simultanément, alors que dans le troisième exemple (c), le processus *C* peut être déclenché soit par *A*, soit par *B*.

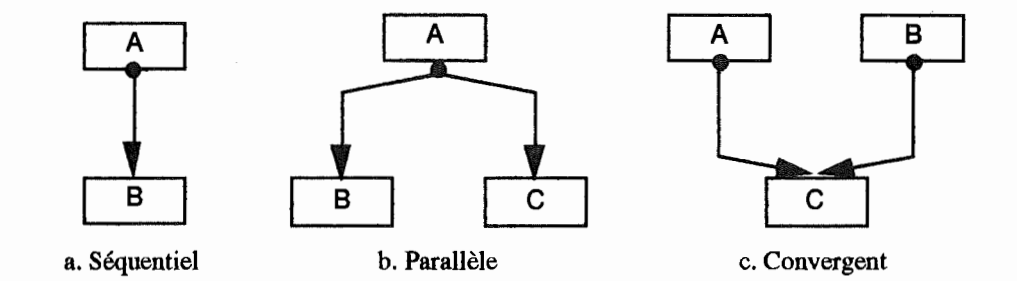


Figure 2.7. Enchaînements dans IDA.

La figure 2.8.a (ci-dessous) montre la représentation graphique d'une synchronisation. Dans cet exemple, le déclenchement du processus *C* nécessite la terminaison des processus *A* et *B* (la condition de synchronisation sera décrite sous forme textuelle).

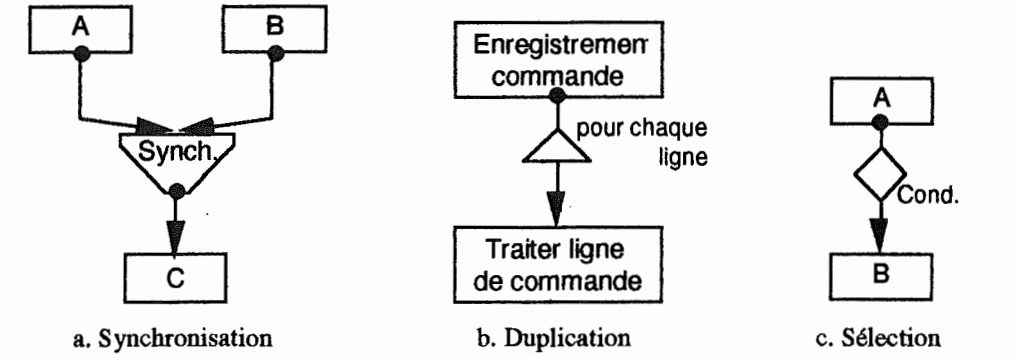


Figure 2.8. Synchronisation, duplication et sélection

Dans l'exemple de la figure 2.8.b, le traitement *Enregistrement commande* déclenche pour chaque ligne de la commande enregistrée l'exécution du traitement *Traiter ligne de commande*. On peut spécifier l'argument de répétition ou le nombre de répétition en l'inscrivant à côté du triangle représentant la duplication : "Pour chaque ligne".

Par contre, le déclenchement du processus *B* ,dans l'exemple 2.8.c, sera soumis à la réalisation de la condition *cond* . Cette dernière sera décrite de manière complète sous forme textuelle.

2.2.5. Modèle des ressources

Ce modèle permet de représenter l'attribution aux traitements des ressources requises pour leur fonctionnement. Ces ressources étant tant humaines que techniques, réutilisables ou non, partageables ou non.

Ce modèle rejoint le modèle organisationnel de MERISE. Cependant, IDA permet également déclarer des règles de priorité et de préemption sur les ressources non partageables, c'est-à-dire qu'un traitement sera prioritaire par rapport à un autre traitement pour l'utilisation d'une même ressource, et qu'il pourra même interrompre (préemption) un autre traitement pour utiliser la ressource utilisée.

Ces règle de priorité et préemption seront explicitées de manière textuelle. Pour chaque ressource on précisera sa capacité maximale et son calendrier de disponibilité. Pour chaque traitement, on spécifiera les ressources requises, la quantité requise, et les éventuels droits de préemption sur d'autres traitements.

2.2.6. Modèle du diagramme des flux

Ce diagramme permet d'avoir une vision globale du système et des flux d'information circulant à l'intérieur de celui-ci et sur ses frontière avec le monde extérieur.

En plus des notions de message, traitement et mémoire déjà rencontrées, on rencontre celles d'unité organisationnelle et d'environnement.

Une unité organisationnelle est un élément de la structure de l'organisation [...] où s'exécutent des traitements et où sont localisés les responsables de leur exécution.

L'environnement du système d'information est constitué de tout ce qui est en dehors du S.I. et qui interagit avec le S.I. par l'échange de messages.

Dans ce modèle, on représente:

- la réception - la génération -, par un traitement, de messages en provenance - à destination - de l'environnement, d'unité organisationnelle ou d'un autre traitement.

- la consultation et la mise-à-jour, par un traitement, de la mémoire du S.I.

De plus, on peut associer aux traitements leur durée d'exécution, et le délai qui les sépare de leur prédécesseur. Aux messages, on associera le nombre d'exemplaires si celui-ci est supérieur à un.

Représentation graphique

Les messages sont représentés de la même façon que dans le modèle de la dynamique.

Les traitements interactifs sont représentés par un trapèze, les traitements entièrement automatisés par un double rectangle, et les traitements totalement manuels par un rectangle simple.

Les flux sont représentés par des flèches et la mémoire du S.I. est représentée par un cylindre.

Les unités organisationnelles correspondent chacune à une colonne du graphique. Dans cette colonne apparaissent les traitements effectués dans cette unité.

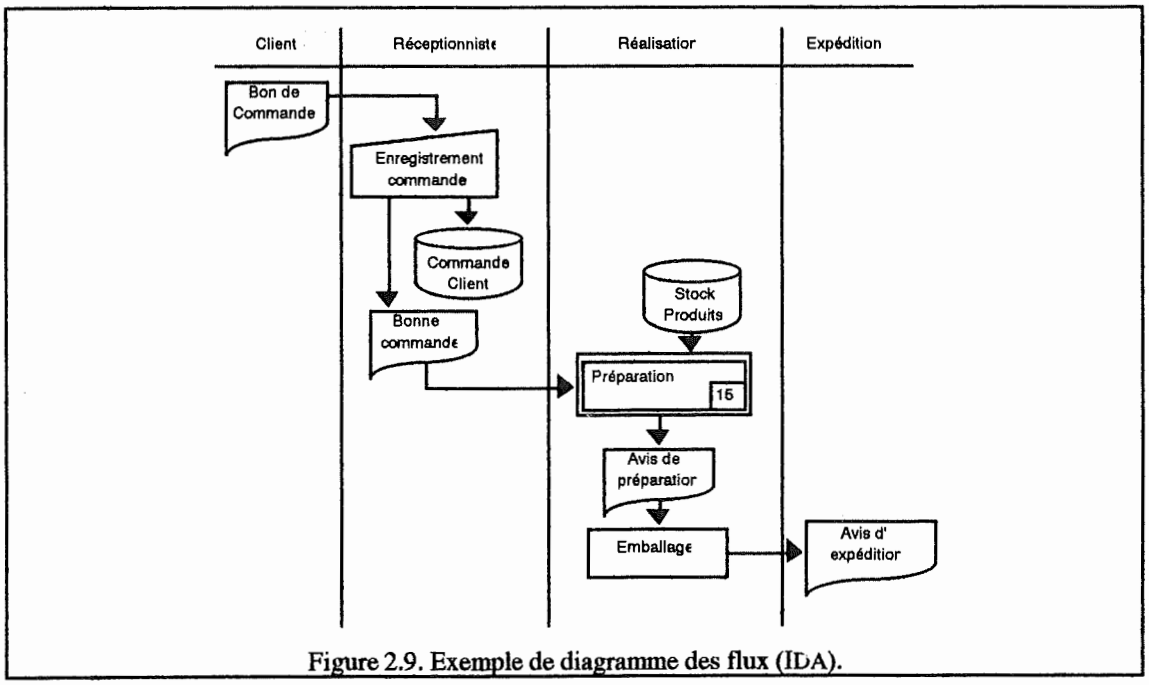


Figure 2.9. Exemple de diagramme des flux (IDA).

2.3. Démarche

La démarche débute par les étapes d'analyse d'opportunité et d'analyse conceptuelle. Elle se poursuit par diverses étapes consacrées à la réalisation technique et à l'élaboration de procédures d'organisation, comme le montre la figure 2.1 (page 19). Seules les deux premières étapes vont être détaillées.

L'analyse d'opportunité

Le but de cette étape est d'élaborer un avant-projet de solution sur base d'un examen approfondi des besoins de l'organisation. On se focalise sur le "pourquoi" des modifications à apporter au système d'information, sur leur cause et leur nature. Cette analyse va se dérouler en diverses phases:

- Identification du projet: il s'agit de délimiter le projet à concevoir en fonction des fonctionnalités attendues, et des objectifs à atteindre. on délimitera également les frontières du projet. Enfin, on spécifiera les différentes applications constitutives du projet.
- Définition du projet-cadre: on exprime les besoins de l'organisation en terme d'objectifs et de critère d'efficacité. On décompose les applications en phases.
- Etude critique de l'existant: afin de fournir des bases factuelles au projet-cadre. On élabore un diagramme des flux pour chaque phase.
- Point de contrôle: vérification des résultats obtenus et confrontation aux attentes de l'organisation.
- Elaboration et évaluation des solutions: Pour chaque phase, chaque application des solutions répondant aux éléments du projet-cadre, on va estimer la rentabilité.
- Choix d'une solution sur base des résultats précédents.
- Point de contrôle.
- Mise-à-jour du projet cadre en fonction de la solution retenue. Le document obtenu devient la référence pour tous.
- Planning des tâches
- Point de contrôle de l'analyse d'opportunité;

L'analyse conceptuelle

Cette analyse se déroule en trois étapes principales (par application):

Elaboration du schéma conceptuel des informations

- Sélection des phases possédant le plus grand contenu informationnel.
- Mise sous forme textuelle du contenu informationnel de ces phases.
- Elaboration d'un schéma entité/association brut.
- Mise sous forme canonique de ce schéma: élimination de redondances éventuelles, stabilisation du schéma.
- Extraction des sous-schéma externes propre à chaque phase
- Point de contrôle.

Elaboration de schéma conceptuel des traitements

Pour chaque phase de chaque application:

- Description des fonctions de la phase: description "statique" : Entrée/sortie, fonctions élémentaires constitutives, description des règles (langage de description DSL[6])
- Réalisation du schéma de la dynamique des fonctions.
- Prototypage de la phase (DSL/PROTO) afin de visualiser le comportement.
- Point de contrôle

Consolidation

- On va vérifier la cohérence globale entre les différentes applications et phases, notamment en ce qui concerne leurs relations (transmission de messages, d'informations). On va également consolider les schémas de données des diverses applications.

- Pour faciliter les choix d'implémentation lors de la réalisation technique, on va quantifier divers éléments tels que:
  - les volumes de données : nombre d'occurrences pour chaque type d'entité et d'association.
  - le nombre d'accès aux informations stockées pour chaque phase.
  - fréquence d'exécution de processus correspondant à chaque phase.
  - nombre et fréquence d'échange de message entre processus.

- Point de contrôle de l'analyse conceptuelle.

---

[6] Dynamic Specification Language. Cf. [BOD. 89] pp. 161 et suiv.

3. Remora

3.1. Introduction

La méthode REMORA est le fruit des travaux menés par C. ROLLAND et son équipe.

La principale caractéristique de REMORA est que l'on ne sépare plus la modélisation des aspects statiques de celles des aspects dynamiques. On inclut les notions d'éléments permanents et d'éléments variables auxquels on associe une date. Cela permet de conserver tous les états successifs du système.

REMORA met l'accent sur la formalisation des divers éléments selon le formalisme relationnel de CODD [COD. 70]. Cela fournit une représentation sous une forme élémentaire et indécomposable, qui permet de tendre vers les objectifs de cohérence, non-redondance et complétude recherché lors de l'étape de conception.

L'exposé de la méthode REMORA suivant est basé principalement sur le livre de C. ROLLAND, O. FOUCAUT, et G. BENCI [ROL. 88], et sur la thèse de O. FOUCAUT [FOU. 82].

3.2. Concepts et Modèles

3.2.1. Modèle descriptif

"Le but du modèle descriptif est de permettre de reproduire la réalité telle qu'elle est en faisant appel à un nombre limité de concepts".

Ces concepts sont les objets, les opérations et les événements.

Les Objets

La description statique des objets, et de leurs inter-relations s'effectue via l'utilisation d'un formalisme entité/relation.

"Un objet est un constituant abstrait ou concret de l'organisation durable dans sa spécificité".

Propriétés d'objet

"Un objet se caractérise des autres objets par des qualités appelées *propriétés* ". Ces propriétés (attributs) portent un nom . Au moins une de ces propriétés est identifiante et garde, tout au long de sa vie, la même valeur.

Association d'objets

Une association est une combinaison d'objets où chacun tient un rôle spécifique. On distingue les associations de composition (un objet est composé d'autres) et de situation (deux objets interviennent dans une situation).

Une association peut posséder des propriétés qui lui sont propres.

Toute association est identifiée par l'ensemble des identifiants des objets qui la composent.

Etats d'objets

Un objet est dans l'état *existant* s'il existe pour l'organisation, c'est-à-dire s'il a subi une opération de création et que l'organisation en a connaissance.

Un objet est dans l'état *inexistant* s'il n'a pas encore été créé, ou que l'organisation n'a pas connaissance de cette création, ou encore, s'il n'existe plus.

Classe d'objets et t-objet

Les objets définis par des propriétés sémantiquement identiques sont regroupés dans des ensembles appelé classe. Une classe est référencée par un nom.

A chaque classe d'objet correspond un objet-type (t-objet) qui permet de définir en compréhension la classe considérée. Chaque objet-type est décrit par des propriétés-types (t-propriétés).

On parlera également de classes d'associations et de t-association.

Les Opérations

"Une opération est une action qui peut être effectuée isolément dans l'organisation et qui modifie l'état de ses objets".Une opération est effectuée par un acteur (humain ou mécanique), dans un lieu et à un moment déterminés, et dure un certain laps de temps.

Propriétés d'opération

Comme les objets, les opérations sont décrites par des propriétés. Ces propriétés sont:

- des propriétés spatio-temporelle (lieu, date, durée de l'exécution).
- des propriétés attachées à son rôle (e.a. le nom du service dont elle dépend).
- les moyens requis (quantité d'énergie, nombre de machines et d'employés, ...), les règles de sécurité, la qualification du personnel nécessaire.

- l'énoncé de la règle d'action (algorithme) sous forme de texte libre.
- éventuellement un identifiant.

Association MODIFIE entre opérations et objets

Les effets d'une opération sur les objets vont être décrits grâce à l'association MODIFIE reliant les objets cibles et l'opération changeant l'état ou les propriétés de ces objets.

On distingue trois type de "modification" opérées sur les objets: la création, la suppression, et la mise à jour (modification des valeurs de certaines propriétés).

On parlera de la classe d'association MODIFIE contenant toutes les associations MODIFIE reliant des objets (ou des associations d'objets) appartenant à des mêmes classes et des opérations d'une même classe.

Association COMPOSE entre opérations

Entre opérations, l'association COMPOSE expriment le fait que l'exécution d'une opération est constituée de la succession d'opérations élémentaires ou elles-mêmes composées.

Classe d'opérations et t-opération

Une classe d'opérations est un ensemble regroupant les opérations qui "modifient des objets ou des associations d'objets appartenant respectivement à des mêmes classes et définies sur des propriétés sémantiquement identiques".

A chaque classe d'opérations, comme on l'a fait pour les classes d'objets, on fait correspondre une opération-type (t-opération)

Les Evénements

Un événement est la constatation du changement d'état d'un ou plusieurs objets. Cette constatation est basée sur un prédicat qui décrit la condition dans laquelle la constatation a lieu.

On remarque que tous les changements d'état d'objets ne sont pas source d'un événement. Seule la constatation de certains d'entre eux est utile à l'organisation.

Propriétés d'événement

Les événements possèdent peu de propriétés descriptives. Ils sont décrits par un identifiant, une date et heure de survenance. Ils sont principalement définis sur base des associations auxquelles ils participent.

Classe d'événements

Une classe d'événements est un ensemble d'événements constatant (Cf. infra) des changements d'état de même nature, d'objets ou d'associations d'objets appartenant respectivement à des mêmes classes déclenchant (Cf. infra) des opérations appartenant à des mêmes classes d'opérations et définis par des propriétés identiques par leur sémantique".

A chaque classe d'événement correspond un événement-type (t-événement) décrit par ses propriétés-types et ses t-associations dont notamment les t-associations DECLENCHE et CONSTATE auxquelles il participe.

Association CONSTATE entre événements et objets.

On retrouve l'association CONSTATE entre l'événement et les objets dont il constate le changement d'état. Le prédicat est une propriété de l'association CONSTATE et non pas de l'événement.

On regroupera les associations CONSTATE portant sur les mêmes classes d'événements et d'objets dans une classe d'associations CONSTATE.

Association DECLENCHE entre événements et opérations.

Un événement peut déclencher une ou plusieurs opérations. L'association DECLENCHE entre événement et opération décrit ce phénomène.

De plus, le déclenchement peut ne survenir que si une condition déterminée est remplie (la condition de déclenchement est une propriété de l'association DECLENCHE). On va de plus préciser les facteurs de déclenchement c'est-à-dire l'ensemble des objets sur lesquels porteront les opérations déclenchées.

Exemple : L'événement *rupture de stock 1234* constate le changement d'état de l'objet "*produit Toto* " (prédicat : quantité disponible inférieure à 20 unités). Cet événement déclenche l'opération "*Commande au fournisseur* " avec le facteur de déclenchement "*produit Toto* ".

On regroupera les associations DECLENCHE portant sur les mêmes classes d'événements et d'opérations dans une classe d'associations DECLENCHE.

Association COMPOSE entre événements.

Un événement peut être la conjonction de plusieurs événements élémentaires ou eux-mêmes composés, se produisant en même temps (en fonction de l'échelle du temps utilisée).

Le modèle descriptif peut être résumé par le schéma suivant:

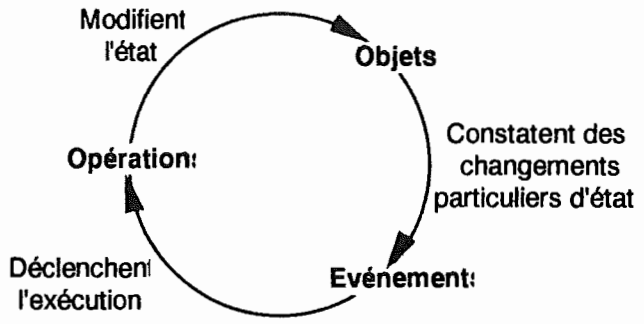


Figure 3.1. Modèle descriptif de la dynamique (REMORA).

3.2.2. Modèle Conceptuel

Le but de ce modèle est de fournir une représentation du système d'information qui soit cohérente, non-redondante, fidèle, complète, simple et lisible.

Après normalisation (mise sous formes normales de Codd), les *t-éléments* deviennent des *c-éléments*.

Le modèle conceptuel contient trois concepts: les c-objets, les c-opérations et les c-événements.

Les c-Objets

"Un c-objet représente un aspect temporel d'un t-objet ou d'une t-association d'objets. Un c-objet est une relation en 3FN permanente qui regroupe un ensemble d'attributs représentant des propriétés du t-objet ou de la t-association qui ont le même comportement au cours du temps".

Les relations de type c-objets ont deux types de clés: les clés contenant des attributs de type Date, et les clés qui n'en contiennent pas.

On a donc deux types de schéma de relation: les relations variables (RV), dont les attributs représentent des propriétés de la classe variables au cours du temps, et les relations permanentes (RP), dont les valeurs des attributs ne changent pas au cours du temps.

Dès lors un t-objet (qu'une t-association) pourra être représenté par la conjonction de plusieurs c-objets (une RP et une ou plusieurs RV). L'ensemble de ces c-objets représentant les différents aspects d'un même t-objet (ou d'une même t-association) sont regroupés dans une c-classe. Cette c-classe permet d'établir aisément un lien entre la représentation conceptuelle et la réalité.

Un c-objet est spécifié par: son nom, son type (permanent ou variable), la liste de ses attributs (avec pour chacun d'eux son nom et son domaine de valeurs), sa clé, le nom de la c-classe à laquelle il appartient, et une liste de contrainte d'intégrité.

Les c-Opérations

"Une c-opération représente un type d'action élémentaire, exécutable dans le système d'information en réponse à un c-événement, agissant sur un unique c-objet".

Toute c-opération porte donc sur un et un seul c-objet, et est associée à l'énoncé d'une règle d'action (sous forme de texte). Elle ne provoque donc que des changements d'état élémentaires d'une seule occurrence d'un c-objet.

Comme pour les c-objets, les c-opérations ont des attributs permanents et d'autres variables. On utilise donc plusieurs types de schéma pour les décrire.

• Un premier type de schéma (nommé ROP) permet d'associer de manière permanente à une c-opération son nom et divers attributs ainsi que le nom du c-objet sur lequel elle porte.

• Le deuxième type de schéma, nommé RTEXT, contient le nom d'une c-opération, une date (constituant à eux deux la clé), et la référence à un texte décrivant les règles de gestion à appliquer.

• Le troisième type de schéma (REXEC) contient l'identifiant d'une c-opération, une date d'exécution de l'opération, et l'identifiant du c-objet sur lequel ont porté les actions associées à l'opération.

Une c-opération est décrite par son nom, le nom du c-objet auquel elle s'applique, l'énoncé de sa règle d'action, et les schémas des relations de type ROP, RTEXT, et REXEC et leur définition[7].

Les c-Evénements

"Un c-événement représente un type de changement d'état élémentaire particulier d'un c-objet qui déclenche l'exécution de c-opérations. La survenance d'un événement (occurrence d'un c-événement) correspond au changement d'état particulier d'une seule occurrence de c-objet et déclenche l'exécution de une ou plusieurs occurrences de c-opérations."

Cinq types de schéma peuvent être utilisés pour décrire les c-événements.

• Le type REV contenant le nom de l'événement, celui de l'objet constaté et ceux des opérations déclenchées, ainsi que divers attributs propres à l'événement.

---

[7] Dans certains cas, les schémas ROP et RTEXT peuvent être regroupés pour alléger les descriptions.

• Le type RPRED permettant de suivre l'évolution du prédicat associé à l'événement.

• Le type RCONDFAC qui permet d'associer un événement, une date et une opération à une condition de déclenchement de l'opération.

• Le type REVARR qui permet d'associer des attributs variables à un événement.

• Et enfin le type REVDECL qui permet de conserver trace du déclenchement d'opérations par un événement.

Un c-événement est décrit par: son nom, le nom du c-objet constaté, la référence du prédicat associé et son énoncé, la liste des c-opérations déclenchées (avec les éventuelles conditions et facteurs de déclenchement).

Le schéma conceptuel

Ce schéma est le résultat de l'étape de modélisation de la réalité. On va y retrouver les "c-éléments" définis ci-avant sous une forme graphique et textuelle, ainsi que l'énoncé de contraintes d'intégrité.

Représentation graphique

On retrouve deux types de graphiques: le premier permet de représenter les c-objets et les c-classes, alors que le second donne un aperçu des liens entre c-objets, c-opération et c-événements.

Représentation des c-objets et c-classes: sous-schéma statique.

Les c-objets sont représentés par un rectangle contenant leur nom et le graphe des dépendances fonctionnelles permanentes entre leurs attributs (Figure 3.2). Les dépendances fonctionnelles sont représentées par les flèches, et leur caractère permanent par les deux points encadrant la flèche.

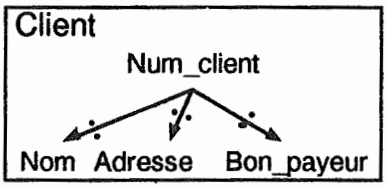


Figure 3.2. Représentation graphique des c-objets (REMORA).

Les c-classes sont représentées par un rectangle, en traits épais, contenant tous les c-objets leur appartenant. La clé du c-objet permanent (de type RP) est la clé de la

classe. On a une dépendance entre la clé de chaque c-objet "variable" et celle du c-objet permanent, appelé racine de la classe.

Représentation des liens entre c-éléments: sous-schéma dynamique.

Les c-éléments sont représentés par un cercle pour le c-objet, une flèche pour la c-opération, et un triangle sur pointe pour le c-événement. La description graphique de la dynamique se fait dès lors aisément.

La juxtaposition d'un cercle et d'un triangle représente la *constatation* d'un objet (cercle) par un c-événement (triangle; Figure 3.3.a). Le fait qu'un c-événement déclenche une ou plusieurs c-opérations se traduit graphiquement par l'association des flèches représentant ces opérations à la pointe du triangle (Figure 3.3.b).

Eventuellement, on peut associer une condition de déclenchement aux opérations en inscrivant l'identifiant de la condition sur la flèche issue du triangle (Figure 3.3.c). Une c-opération ne modifiant qu'un seul c-objet, la flèche associée aboutira sur un seul cercle. De plus, lorsque le déclenchement est itératif, la flèche porte une double pointe marquée par le nom du facteur itératif (Figure 3.3.d).

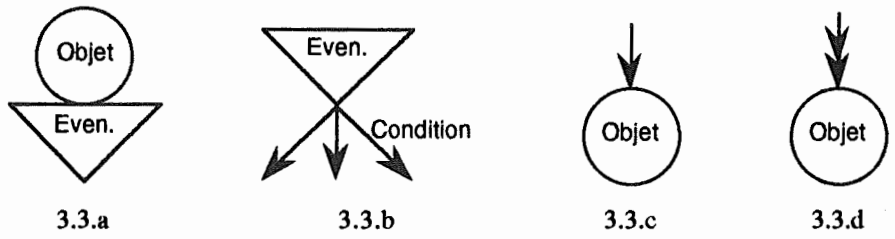


Figure 3.3. Les "associations" graphiques utilisées (REMORA).

L'unité de base du graphique ainsi obtenu, c'est-à-dire la suite d'un triangle, de flèches, et de cercles, est appelée cycle dynamique.

Chaque schéma sera accompagné d'un inventaire reprenant toutes les abréviations employées dans les schémas et leur signification (description).

Contraintes d'intégrité

Les contraintes d'intégrité portent sur les éléments de la base de données, c'est-à-dire sur les c-objets.

Elles ne sont pas représentées dans les schémas graphiques. Elles sont reprises sous forme d'un texte (suite de phrases) joint au schéma graphique.

Description textuelle

Il s'agit d'une description textuelle des éléments contenus dans les schémas graphiques. Cette description est plus complète notamment en ce qui concerne les règles d'action associées aux c-opérations.

REMORA propose un langage appelé REMO-LANGUE, dont la syntaxe est assez proche de SQL.

3.2.3. Modèle logique

Ce modèle logique a pour but de fournir une solution technique pour le système d'information. Cette solution sera constituée d'une base de données et d'un ensemble de programmes.

Sous-schéma logique des données

En ce qui concerne la base de données, elle sera décrite par le "sous-schéma logique des données" dont le contenu est directement dérivable du modèle conceptuel..

Une fois que les éléments du schéma descriptif ont été mis sous forme canonique, on est assuré de les avoir tous représenté de manière adéquate, non redondante.

Cette représentation sous forme de relations (au sens du modèle relationnel de Codd) est appelé "structure logique maximale standard".

Cette structure maximale peut ne pas être adaptée en fonction des utilisations que l'on compte en faire. On va donc la modifier pour améliorer les temps d'accès, optimiser l'exploitation de la base de données.

On obtient une "structure logique adaptée standard".

En fonction du gestionnaire de base de données disponible, il va falloir adapter cette structure pour obtenir une structure logique spécifique.

Sous-schéma logique transactionnel

On va également définir le "sous-schéma logique transactionnel", dérivé du modèle conceptuel et basé sur les notions de transaction et de déclencheur.

Les Transactions

Une transaction est une suite d'opérations atomiques qui fait passer la base de données d'un état cohérent à un autre. Les opérations d'une transaction devront toutes être exécutées ou aucune ne devra l'être.

Chaque transaction n'a qu'un seul point d'entrée, elle a des paramètres en entrée et en sortie. Une transaction sera donc l'équivalent d'une procédure en programmation classique.

Les déclencheurs

Un déclencheur est un prédicat sur l'état de la base de données. Ce prédicat indique quelle condition doit être vérifiée pour déclencher la transaction associée.

Un déclencheur ne peut démarrer qu'une seule transaction, mais une transaction peut contribuer à la réalisation du prédicat de plusieurs déclencheurs.

De même, plusieurs transactions peuvent réaliser le prédicat d'un même déclencheur. Une transaction peut être itérative, c'est-à-dire qu'elle génère plusieurs ensemble de valeurs vérifiant le prédicat du déclencheur.

Schéma transactionnel adapté

A l'instar de la structure logique des données, on vise, en fin du processus de modélisation, à réaliser un schéma transactionnel dit *adapté*.

Ce schéma sera conforme non seulement aux désirs du concepteur et de l'utilisateur , mais également adapté à la modélisation qui a été faite de la base de donnée.

3.3. Démarche

La démarche proposée se déroule en deux phases.

3.3.1. La phase conceptuelle

Durant cette phase, on va décrire, tout d'abord, le système d'information tel qu'il est perçu par les concepteurs: élaboration du modèle descriptif et création des t-éléments.

Ensuite, on va normaliser les t-éléments ce qui mènera à la création de c-éléments,

Enfin aura lieu une étape de contrôle et de validation du travail effectué lors de cette phase. Le schéma conceptuel obtenu est le résultat de cette première phase.

3.3.2. La phase logique

Cette seconde phase sera une étape de modélisation logique qui visera l'obtention d'une solution technique intégrée.

On va obtenir cette solution technique en construisant une structure logique de données.

Ensuite, une structure transactionnelle sera dérivée des éléments conceptuels et adaptée aux besoins des utilisateurs en fonction des possibilités techniques du moment.

---

4. Object-Oriented Analysis

4.1. Introduction

Le fil conducteur de cette méthode est d'aborder le problème de la modélisation d'un système d'information par les données à manipuler.

Cette façon de considérer le problème est justifiée par la constance des informations manipulées, par opposition aux changements subis par les opérations à mener sur ces informations. « Les besoins essentiels d'un système sont caractérisés comme étant les besoins qui subsisteront dans un environnement de technologie d'implémentation parfait » [BLA. 90 p. 23].

L'élaboration d'un système d'information par la méthode OOA permet au système de bénéficier de divers avantages:

• Seules les opérations définies sur un objet peuvent accéder à la structure interne de cet objet. Les opérations sont déclenchées par l'envoi d'un message à l'objet. Le principal avantage qui en découle est que l'implémentation des objets est indépendante de celle des programmes qui les utilisent.

• Comme différentes classes d'objets peuvent répondre différemment à un même message, il est possible d'ajouter de nouvelles classes d'objets pour faire évoluer les applications vers d'autres domaines.

• Il est possible de définir une nouvelle classe d'objets sur base d'une classe pré-existante, sans avoir à redéfinir les similitudes entre ces deux classes. De plus, cette nouvelle classe hérite également des opérations définies sur la classe pré-existante.

La description de la méthode OOA présentée ci-dessous est principalement basée sur l'ouvrage de S. SHLAER et S.J. MELLOR [SHL. 88].

4.2. Modèles & concepts

La méthode OOA propose trois modèles qui sont le "modèle d'information" (*Information Model*),le "modèle d'états" (*State Model*), et le "modèle des traitements" (*Process Model*).

4.2.1. L'*Information Model*

Ce modèle reprend un modèle graphique intitulé "Information Structure Diagram", ainsi que deux documents textuels: l' "Object Specification Document" et le "Relationship Specification Document".

Trois concepts de base sont utilisés dans l'*Information Model* : les objets, les attributs et les relations.

Les objets

Un objet est "une abstraction d'un ensemble de choses du monde réel telle que: - toutes les choses de cet ensemble possèdent les mêmes caractéristiques
- toutes les choses de cet ensemble respectent et sont régies par les mêmes règles"[SHL. 88 p.14].

OOA définit également les notions de sous-type et de sur-type. Cela permet la représentation de similitudes observées dans le monde réel entre deux objets conceptuellement distincts. Par exemple, dans un hôpital, les patients et les employés ont des caractéristiques communes (nom, prénom, date de naissance,...). Ici on peut créer un sur-type "personne" et deux sous-types "patient" et "employé".

Les attributs

Un attribut est "l'abstraction d'une seule caractéristique possédée par toutes les entités, elles-mêmes considérées comme un *objet* "[SHL. 88 p.26].

On retrouve également les notions d'identifiants simples ou multiples.

Il existe trois types d'attributs: les attributs descriptifs, nominatifs et référentiels.

- Un attribut descriptif donne une caractéristique de l'objet. Les attributs "couleur", "vitesse de croisière" de l'objet "véhicule" sont descriptifs.
- Un attribut nominatif contient un libellé arbitraire associé à une occurrence de l'objet, ne reflétant aucune caractéristique de l'objet du monde réel. L'attribut "numéro de patient" de l'objet "patient" est un attribut nominatif.
- Un attribut référentiel est un attribut qui lie une occurrence d'un objet à une occurrence d'un autre objet. L'attribut "numéro de chauffeur" de l'objet "véhicule", référençant une occurrence de l'objet "chauffeur", est un attribut référentiel. On parle aussi de clé étrangère.

Une fois définis, les objets vont être mis sous forme "canonique", c'est-à-dire en troisième forme normale telle que définie par CODD [COD. 70].

Les relations

Une relation est l'abstraction d'un ensemble d'associations qui ont lieu systématiquement entre différentes choses du monde réel.

- Les relations binaires inconditionnelles:

- Relation (1:1) qui à toute occurrence a de l'objet A fait correspondre une et une seule occurrence b de l'objet B, et réciproquement.
- Relation (1:M) qui à toute occurrence de l'objet A fait correspondre au moins une occurrence de l'objet B (relation "One to Many"), et à toute occurrence de B fait correspondre une et une seule occurrence de A.
- Relation (M:M) qui à toute occurrence de l'objet A fait correspondre au moins une occurrence de l'objet B (relation "One to Many"), et à toute occurrence de B fait correspondre au moins une occurrence de A.

• Les relations binaires conditionnelles:

Il s'agit ici de sept relations, dérivées des trois types inconditionnels, pour lesquelles la participation d'occurrence d'objet est régie par des conditions. Cela signifie que certaines occurrences d'objets ne participent pas à la relation.

Le tableau (4.1) ci-dessous résume les différents types de relations binaires, compte tenu que les ensembles représentent des objets, et leurs éléments des occurrences de ces objets, et que la lettre "c" représente la condition.

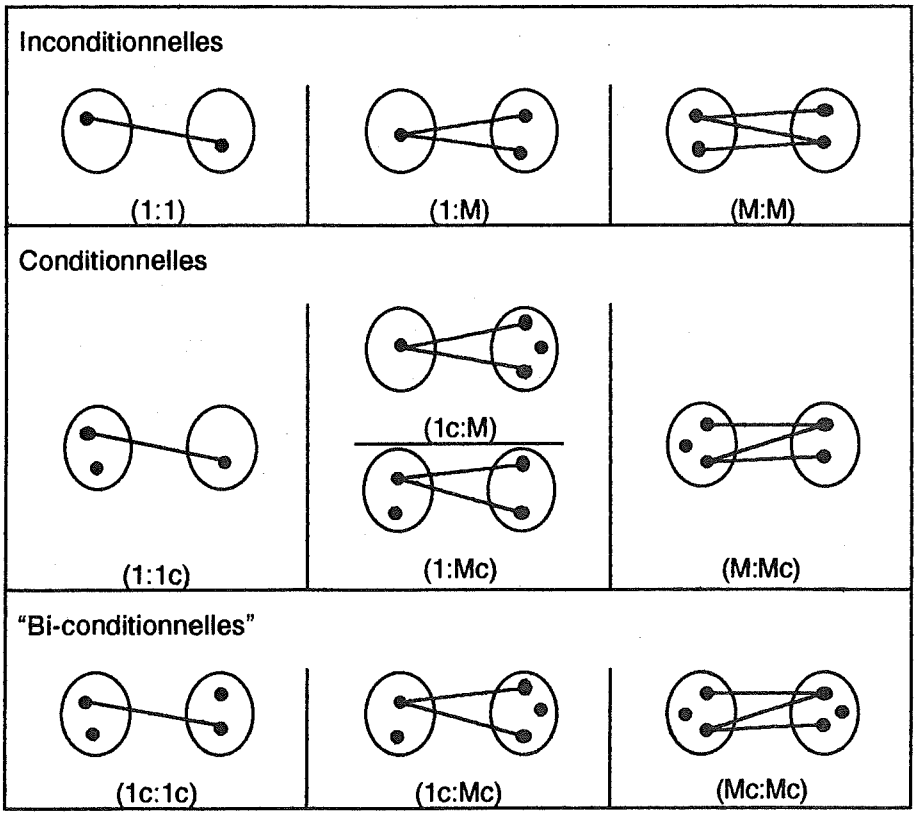


Tableau 4.1. Les différents types de relations binaires.

La modélisation des relations (mise sous forme normale de CODD des objets) va engendrer la création de trois nouveaux éléments: les attributs référentiels (déjà évoqués), les tables de corrélation et les objets associatifs.

• Les tables de corrélation

Les tables de corrélation apparaissent lorsque l'on veut modéliser une relation de la forme (M:M).

Par exemple, on dispose de deux objets: Pièce et Appareil. On établit la relation "est composante de" entre ces deux objets (Figure 4.1). Il s'agit bien d'une relation (M:M) si l'on considère qu'une pièce est composante d'au moins un appareil, et chaque appareil est composé d'au moins une pièce.

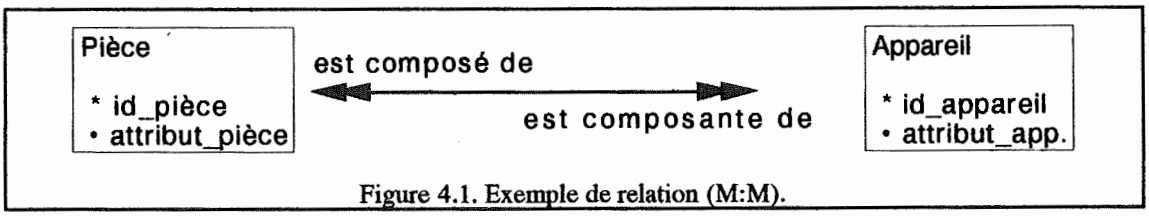


Figure 4.1. Exemple de relation (M:M).

Pour conserver les relations établies entre les occurrences des deux objets, il nous faut une table contenant, pour chaque occurrence de la relation, les identifiants de chacun des objets reliés. Cette table est appelée table de corrélation.

Pour chaque occurrence de la relation "est composante de", la table contient une et une seule ligne de référence.

Ce mécanisme sert également dans les cas de modélisation de relation conditionnelle.

• Les objets associatifs

Un objet associatif est créé lorsque des informations sont associées aux occurrences d'une relation, c'est-à-dire lorsque la relation à modéliser possède des attributs.

Contrairement aux table de corrélation, il est possible avec un objet associatif d'avoir plusieurs occurrences de cet objet associées à une seule occurrence de la relation. On a dès lors une relation "M-(X:Y)" plutôt que "1-(X:Y)", avec X et Y valant 1 ou M.

Par exemple, dans une entreprise de Taxi, les chauffeurs peuvent conduire des plusieurs véhicules. On a une relation (M:M) entre **chauffeurs** et **véhicules.**

Si on désire garder trace des jours où un chauffeur conduit un véhicule, on va associer à la relation **conduire** un attribut **date.** On aura, pour modéliser cette

relation, un objet associatif contenant l'identifiant du chauffeur, celui du véhicule et une date.

Pour un chauffeur déterminé, une liste de véhicules qu'il a conduits, et les dates associées. Pour un chauffeur et un véhicule déterminés, on pourra avoir plusieurs dates associées.

• Les relation ternaires et supérieures

Ces relations sont semblables aux relations binaires. Lorsque trois objets sont associés par une telle relation, les flèches représentant la relation partent d'un losange central, et se dirigent vers les objets. Les notions de table de corrélation et d'objet associatif sont conservées. On remarquera simplement que, graphiquement, toute flèche pointant le losange central est toujours celle d'un objet associatif ou d'une table de corrélation.

L'*Information Structure Diagram*

Ce diagramme constitue la première des trois parties de l'*Information Model*. Il contient une représentation graphique proche des diagrammes "Entités-Relations". On va y retrouver les différents concepts définis ci-dessus.

Les objets seront représentés par un rectangle contenant leur nom.

Les "sur-type" et "sous-types" sont représentés comme des objets normaux et sont reliés entre eux comme montré sur l'exemple ci-dessous (Figure 4.2). La petite barre horizontale est placée sur la branche reliant le sur-type aux sous-types associés.

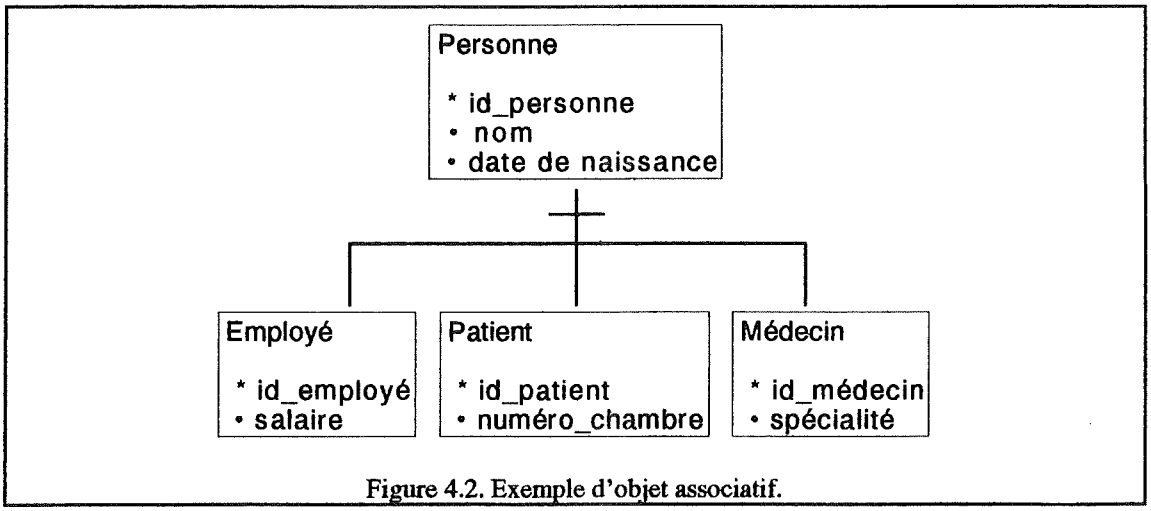


Figure 4.2. Exemple d'objet associatif.

Les attributs, s'ils sont représentés, se trouveront dans le rectangle de l'objet auxquels ils sont associés, sous le nom de cet objet (Cf. figure 4.3 ci-dessous). Une

astérisque précédent le nom d'un attribut signifie qu'il est identifiant (ou partie de l'identifiant si plusieurs attributs sont précédé d'une astérisque).

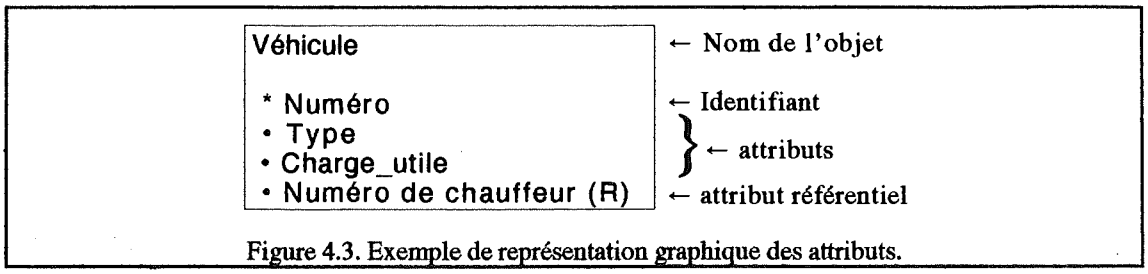


Figure 4.3. Exemple de représentation graphique des attributs.

Les relations sont représentées par une flèche selon les correspondances représentées dans le tableau suivant:

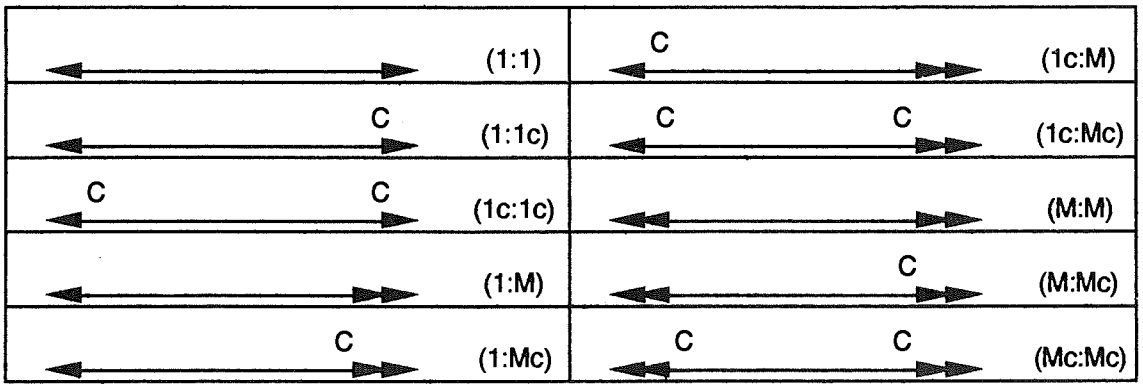


Tableau 4.2. Représentation graphique des relations

L'intitulé des relations se place au bout de la flèche. Ainsi dans l'exemple de la figure 4.1, un Appareil "est composé de" Pièce, et une "Pièce" "est composante de" Appareil. Dans le cas d'une relation conditionnelle, la lettre "c" est placée sur la flèche.

Les tables de corrélation sont représentées par un rectangle, contenant les attributs identifiants des divers objets mis en relation, d'où part une flèche, à pointe simple, en direction du losange symbolisant la jonction entre la relation et cette table (Figure 4.4 ci-dessous).

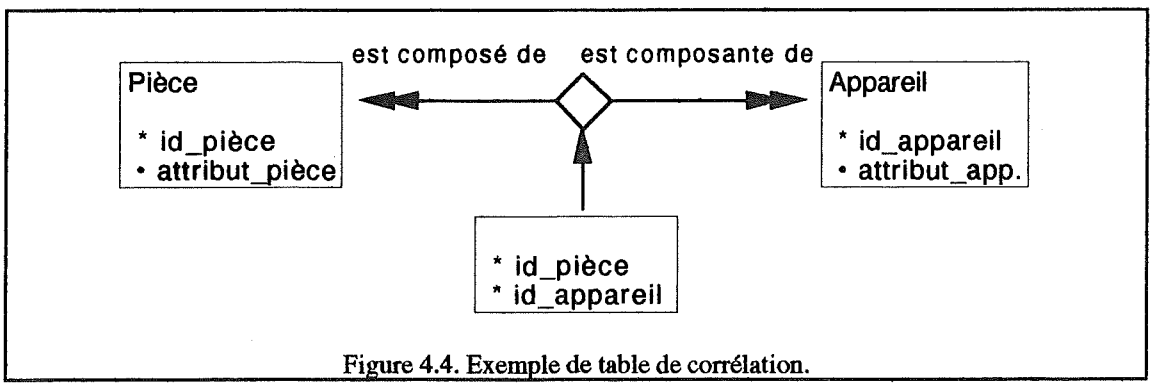


Figure 4.4. Exemple de table de corrélation.

Un objet associatif est représenté comme une table de corrélation avec en plus le nom de l'objet dans le rectangle (Figure 4.5 ci-dessous). Une flèche à double pointe en direction de la relation représente une relation "M-(X:Y)", avec (X:Y) la multiplicité de la relation.

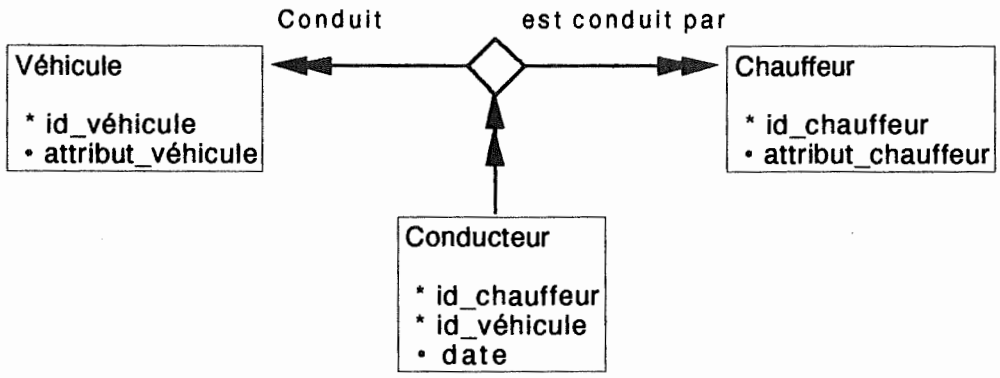


Figure 4.5. Exemple d'objet associatif, et de relation M-(M:M).

Il est possible de représenter le fait que toutes les occurrences d'un objet doivent être reliées à toutes les occurrences d'un autre objet. Cela se fait via une relation (M:M) sur laquelle on place un losange contenant une croix (Figure 4.6).

Figure 4.6. Exemple de relation (M:M) "totale".

L'association de tous ces éléments graphiques permet d'obtenir une vue d'ensemble du problème. Cependant, cette vue n'est pas suffisante, et il est nécessaire de la compléter par des définitions textuelles. C'est l'objet des deux documents suivants: "l'Object Specification Document" et le "Relationship Specification Document".

l'*Object Specification Document*

Il s'agit d'un document textuel donnant une description précise des objets et de leurs attributs.

Pour chaque objet, ce document contient:

- Le nom de l'objet
- La déclaration de l'objet et de ses attributs (les attributs identifiants étant soulignés) sous forme d'une liste: Objet (identifiant, attribut_1,attribut_2,...).
- La liste des identifiants (redondance volontaire par rapport à la déclaration).

- La description de l'objet. Il s'agit d'un texte faisant le lien entre l'objet et le monde réel.
- Pour chaque attribut de l'objet :
  - Son nom.
  - Sa description textuelle.
  - Son objectif, son but. Modélise-t-il une relation et laquelle.
  - Son domaine de définition: l'ensemble des valeurs qu'il peut prendre.

On définira de même les objets associatifs, mais pas les tables de corrélations.

Le *Relationship Specification Document*

Ce document décrit les relations entre objets. Pour chaque relation, on aura:

- Le nom de la relation, les objets qu'elle relie, la multiplicité et la "conditionnalité" sous la forme (1:Mc),(1:1),...
- Une description de la relation du monde réel modélisée avec une justification de la multiplicité et de la conditionnalité.
- Une description de la formalisation de la relation par des attributs référentiels ou des tables de corrélations.

Résumé de l'Information Model

Ce dernier document de l'Information Model, reprend tous les éléments définis auparavant sous une forme simplifiée. Les objets seront repris sous la forme suivante: Le nom de l'objet suivi de, entre parenthèses, le nom de ses identifiants et de ses attributs. Les relations sont quant à elles reprises sous la forme: Nom de la relation, liste des objets reliés et cardinalité sous la forme (X:X[c]) où X vaut "1" ou "M", et la conditionnalité est représentée par la lettre "c" si nécessaire.

Une fois l'Information Model réalisé, on va réaliser le modèle d'états.

4.2.2. "Modèle d'états"

Certains objets évoluent selon un "cycle de vie" qui leur est propre. Dans ce cycle de vie, une occurrence de l'objet passe par différentes étapes appelées *états* .

Par exemple, l'objet "Compte" possèdera notamment les états suivant: état "Inexistant", état "Solde positif", état "Solde négatif", ...

On va ajouter à chaque objet concerné, un attribut, généralement intitulé "status", dont le domaine de définition est la liste des états par lesquels transite cet objet.

On observe qu'un objet passe d'un état à un autre sous l'effet d'une *transition* provoquée par l'occurrence d'un *événement*.

Enfin, des *opérations* peuvent être associées à un état. Lorsqu'une occurrence d'un objet passe dans cet état, les opérations associées sont effectuées.

Une opération peut générer un événement à destination du même objet ou d'un autre. Cela permet de réaliser la "coordination d'état" nécessaire afin de respecter la multiplicité/conditionnalité des relations.

Par exemple, le passage à l'état "Existant" d'une occurrence de l'objet "Client" va générer l'événement "Nouveau compte" à destination de l'objet "Compte". Cet événement va engendrer le passage d'une occurrence de l'objet "Compte" à l'état "Existant". Cela permet de conserver la relation (1:M) entre les objets "Client" et "Compte".

State Transition Diagram - State Model

Ce diagramme permet d'avoir une vue schématique des états par lesquels passent les occurrences d'un objet, ainsi que des transitions et événements. Un tel schéma est réalisé pour chaque objet qui possède un "cycle de vie".

Un état est représenté par un rectangle contenant son nom.

Une transition est représentée par une flèche à laquelle on adjoint le nom de l'événement.

Les événements sont repris dans une table donnant leur description et les données l'accompagnant éventuellement.

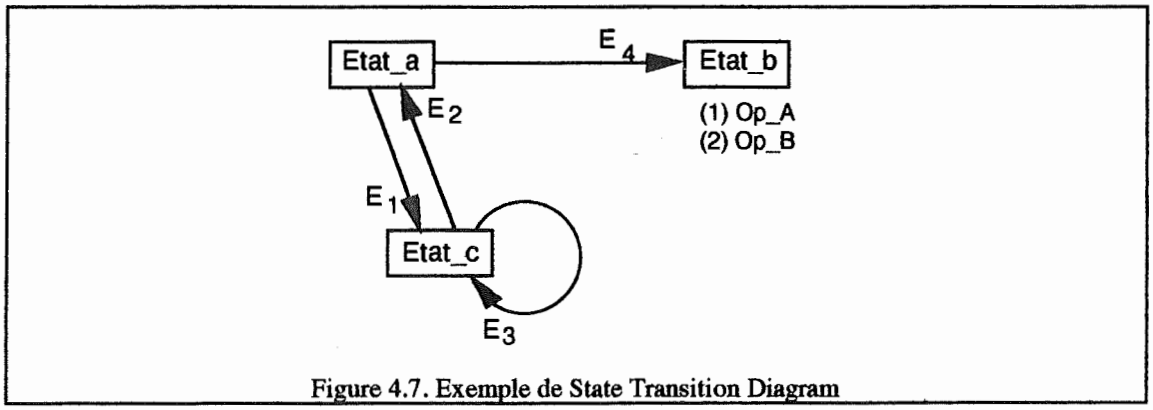


Figure 4.7. Exemple de State Transition Diagram

Les opérations associées à un état sont transcrites en-dessous du rectangle représentant l'état. Pour des raisons de commodité et de clarté du schéma, il peut s'agir simplement de références à ces opérations. Les opérations seront décrites de manière détaillée dans le diagramme des traitements.

Dans l'exemple ci-dessus (Figure 4.7), l'événement "E_1" fait passer l'objet de l'état "Etat_a" à l'état "Etat_c". Les opérations "Op_A" et "Op_B" sont exécutées, dans l'ordre indiqué, lorsque l'objet passe dans l'état "Etat_b".

4.2.3. Modèle des traitements (*Process Model*)

Les transformations et les manipulations effectuées par les traitements sur les données vont être représentées grâce à un diagramme de flux de données ou Data Flow Diagram [GAN. 79], [ROS. 77 a,b].

Data Flow Diagram : DFD

Les éléments constitutifs d'un DFD sont:

- Les **entités externes** qui sont sources ou destination du flux de données. Par exemple : les clients, les fournisseurs.
- Les **fichiers de données.**: ils contiennent les informations manipulées par le système. On remarque que les fichiers de données sont directement dérivés des objets de l'*Information Model* .
- Les **opérations** (ou fonction, processus) qui transforment les données. Lorsqu'un flux de données arrive à une opération, cette dernière s'effectue sur base des informations contenues dans le flux.
- Les **flux de données** circulant dans l'organisation.

Ces quatres éléments sont les briques de base de tout diagramme de flux de données. Ils permettent de décrire un système d'information à des niveaux de précision différents.

La description des opérations se fera sous forme textuelle. Cependant, il peut se faire qu'une opération soit trop complexe que pour être aisément décrite par un texte simple.

Dès lors, on va la décrire par un DFD "fils" du DFD général. Ce nouveau DFD contiendra des opérations qui, éventuellement, pourront également être décomposées. Cette technique permet de garder au DFD général une certaine simplicité et donc une meilleure communicabilité.

Dans les diagrammes de haut-niveau, on ne tient pas compte des traitements d'exception (traitement d'erreur),et on ne distingue pas les opérations manuelles des opérations informatisées. Cette distinction sera faite sur un DFD de niveau assez détaillé. En règle générale, le sous-système à informatiser sera entouré d'un trait en pointillé sur le DFD. Ainsi, les limites du sous-système informatique sont précises. On détaillera ensuite ce sous-système en réalisant pour chaque opération un nouveau DFD.

Représentation graphique

Malgré des concepts de base identiques, plusieurs formalismes graphiques existent pour représenter les éléments d'un DFD.

Le tableau 4.3 ci-dessous montre les deux formalismes graphiques les plus employés [MAR. 78], [GAN. 79].

Eléments	Formalisme de YOURDON / De MARCO	Formalisme de GANE & SARSON
Opération (fonction)	Nom de la Fonction	Nom de la Fonction
Fichier de données	Nom des données	# Nom des données
Agent externe	Nom de l'agent	Nom de l'agent
Agent interne	Nom de l'agent	Nom de l'agent
Flux de données	Nom du flux →	Nom du flux →

Tableau 4.3. Formalisme des diagrammes de flux de données.

4.3. Démarche

La démarche proposée est, somme toute, classique. Elle se déroule en quatre phases qui sont l'analyse du problème, les spécifications externes, le "design" du système, et, enfin, l'implémentation. C'est par l'approche utilisée dans la première phase que se distingue la méthode OOA vis-à-vis des autres méthodes.

4.3.1. La phase d'analyse

L'objectif de cette phase est d'élaborer trois modèles à partir des informations de bases. Ces modèles sont le "modèle d'information" (*Information Model*),le "modèle d'états" (*State Model*), et le "modèle des traitements" (*Process Model*). La création des trois modèles s'enchaînent comme le montre la figure 4.8 ci-dessous.

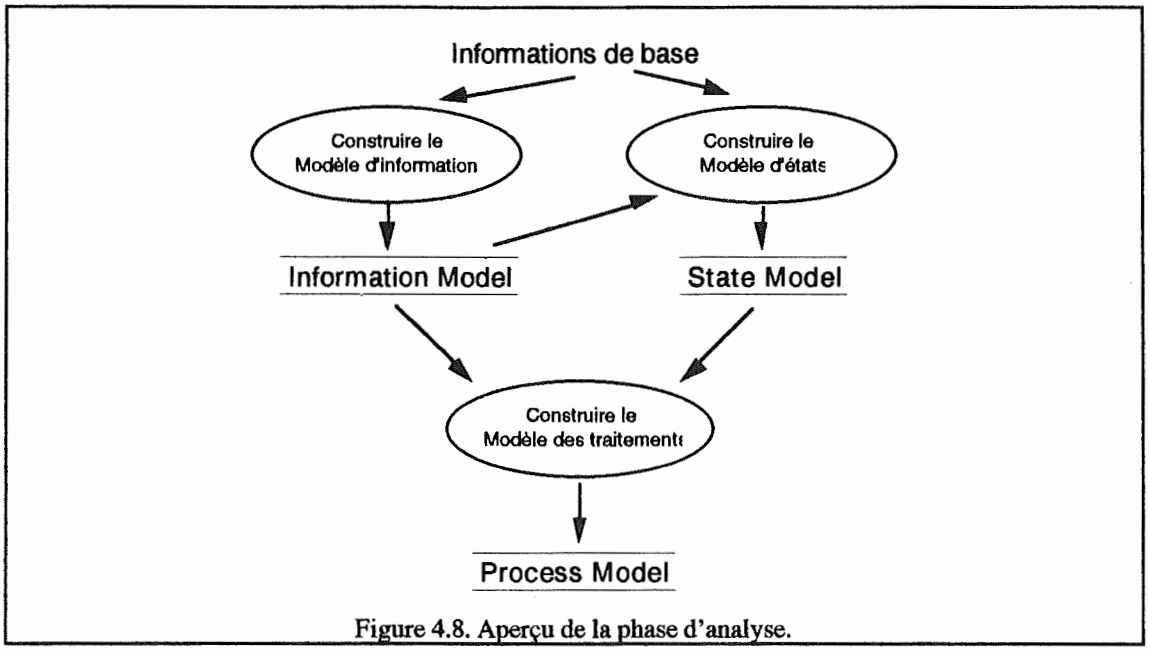


Figure 4.8. Aperçu de la phase d'analyse.

4.3.2. Phase de spécification externe

Cette seconde phase à pour but de déterminer la répartition des rôles entre machines, opérateurs et employés.

Pour ce faire, on doit d'abord déterminer les opérations qui seront automatisées, et celles qui resteront manuelles. La partie à automatiser devient dès lors, pour le reste de cette phase, une "boîte noire".

Sur base des trois modèles réalisés lors de la première phase, on va élaborer l'***external event list*** c'est-à-dire la liste des événements qui proviennent de l'extérieur vers le système automatisé. Pour chacun de ces événements, on va reprendre son nom, sa définition et son contenu.

Ensuite, on va élaborer les ***Narrative Requirements Documents*** qui reprennent ce que le système doit faire en réaction aux événements reçus (spécifications fonctionnelles). Ces documents sont écrits en langage naturel sur base des informations et définitions fournies par les résultats de la phase d'analyse.

Enfin, on décrit l'interface entre le système automatisé et les utilisateurs (écrans, rapport, procédures que doivent suivre les opérateurs), ainsi que les éventuelles relations entre le système automatisé et d'autres systèmes.

4.3.3. Phase de Design

Software architecture design

On va définir les accès, et leur organisation, aux données partagées. Les modes de déclenchement des programmes, ainsi que d'autres conventions telles que les interfaces entre les différents programmes.

Design of system information content

A partir du modèle d'information, on ne va conserver que les informations réellement utilisées par le système automatisé. Cela va produire un modèle d'information "réduit".

Data structure design

Sur base du modèle d'information réduit, du sens de parcours des relations entre objets, et des fréquences d'accès aux données, on dérive la structure des données à implémenter.

Découpe en programmes

Il s'agit de répartir les différentes opérations à réaliser en différents programmes, en tentant d'optimiser les aspects suivants: clarté, ré-utilisabilité, structuration du code, indépendance des procédures vis-à-vis du programme global,...

4.3.4. Phase d'implémentation

Sur base des résultats des phases précédentes, il s'agit maintenant de réaliser les programmes, en fonction des moyens techniques disponibles.

5. SADT

5.1. Introduction

"Structured Analysis & Design Techniques"[8] est une méthode d'analyse développée par Douglas T. ROSS [ROS. 77], [LIS. 90]. Il ne s'agit pas, à proprement parler d'une méthode de conception de système d'information: SADT n'aborde que les problèmes de modélisation rencontrés lors de l'analyse fonctionnelle, et cette méthode pourrait être employée lors de tout développement.

Pour la phase d'analyse fonctionnelle, SADT offre les techniques et les méthodes pour analyser et modéliser les problèmes d'un point de vue structuré, et pour planifier et gérer le travail d'équipe (répartition, coordination, rôles, documents).

La première caractéristique de SADT est la décomposition hiérarchique du système. Un système est d'abord décrit par une boîte noire, qui ne révèle rien de son contenu. On en donne juste le nom, et les relations que cette boîte entretient avec son environnement.

Le contenu de cette boîte va, ensuite, être détaillé par un diagramme composé d'autres boîtes et d'autres flèches représentant les relations entre ces boîtes. Chacune de ces boîtes va à son tour être détaillée... Et ce jusqu'au moment où le niveau de détail atteint est jugé suffisant.

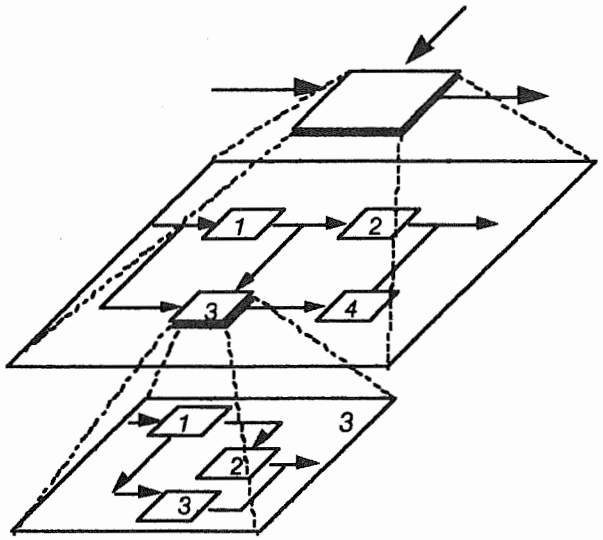


Figure 5.1 : Décomposition hiérarchique dans SADT [ROS. 77 b]

[8] SADT est une marque déposée de SOFTECH Inc.

Pour conserver la clarté d'une telle hiérarchie, SADT impose la règle suivante: une boîte mère ne peut avoir moins de trois boîtes filles (afin de garantir une utilité à la décomposition), et pas plus de six (pour la clarté du schéma).

Cette décomposition hiérarchique était déjà présente dans les diagrammes de flux de données (Cf. 4.2.3).

Aux *Data Flow Diagrams* définis par la méthode *Structured Systems Analysis* [GAN. 79], SADT apporte deux modifications majeures:

- Le formalisme proposé dans SADT permet de différencier le rôle joué par les flux de données: entrée, sortie, contrôle, ou mécanisme.
- Sur base du même formalisme (boîtes & flèches), SADT permet de mettre en évidence la dualité activités - données de tout système d'information.

5.2. Modèles et concepts

5.2.1. Dualité activités - données

Quel que soit le système à réaliser, il sera décrit en terme d'activités et de données. Dans un formalisme textuel, cette dualité activités - données se traduirait par l'utilisation de verbes et de noms.

Se limiter à la description d'un seul de ces deux aspects, tronquerait la vision de la réalité.

Dans SADT, les aspects "données" et "activités" seront toujours représentés simultanément. Cependant, pour des raisons de facilité de modélisation, on mettra l'accent sur un seul de ces aspects. Cela va mener à la création de deux approches:

- La première sera basée sur les activités (*ActiGrams* ou *Activity-Diagram*),
- La seconde sur les données (*DataGrams* ou *Data-Diagram*).

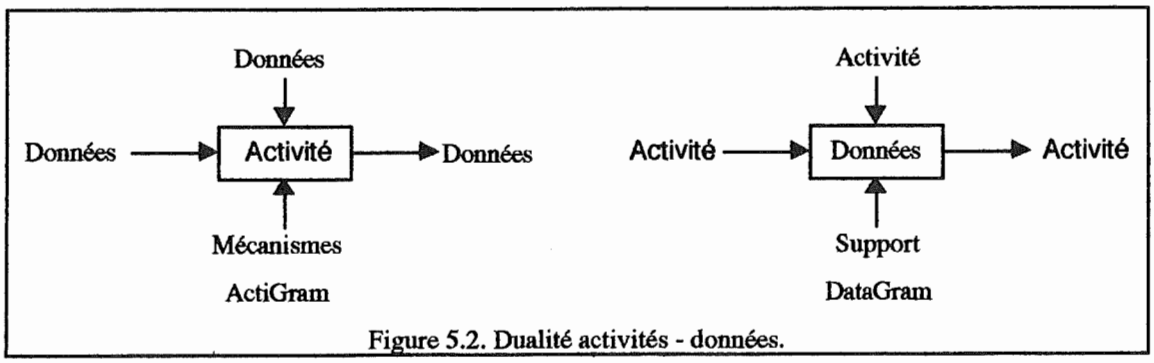


Figure 5.2. Dualité activités - données.

Ces deux approches seront concrétisées par la réalisation de diagrammes basés sur le même formalisme graphique (Cf. figure 5.2 & infra].

L'étape finale de toute conception consistera donc à confronter les différents diagrammes, et à vérifier que chaque élément (boîte ou flèche) se retrouve sous l'autre forme (respectivement flèche ou boîte) dans l'autre famille de diagrammes.

5.2.2. Les ActiGrams

Les boîtes

Dans un ActiGram, les boîtes représentent les activités.

Chaque activité est décrite par un verbe accompagné éventuellement d'un commentaire. On associe également à chaque activité sa ou ses équations d'activation (Cf. infra: données de contrôle).

S'il est suffisamment complexe (au moins trois sous-activités), le contenu d'une activité pourra être décrit par un autre ActiGram, plutôt que de manière textuelle.

Les flèches

Les activités peuvent recevoir des données en entrée, les transformer par l'utilisation de mécanismes sur base de données de contrôle, et produire des données de sortie.

Les flèches représentent ces flux de données entre activités. On attribue à chaque flèche une étiquette référençant les informations contenues dans le flux représenté.

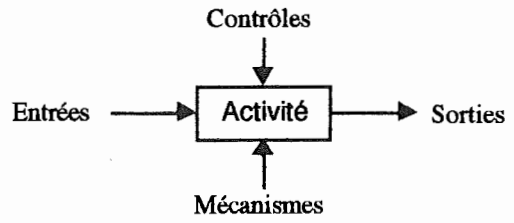


Figure 5.3. Eléments de base des ActiGrams.

Les données en entrée

Ces données proviennent soit:

- d'une autre activité appartenant au même diagramme
- du contexte de la boîte mère, c'est-à-dire qu'il s'agit de données (en entrée ou de contrôle) arrivant à l'activité mère.

Il faut remarquer que l'arrivée de données en entrée ne déclenche pas l'activité. Pour ce faire, il faudra attendre l'arrivée de données de contrôle.

Graphiquement, les données en entrée sont représentées par les flèches arrivant au côté gauche de la boîte.

Les données de contrôle

Les données de contrôle, parvenant à une activité, ne sont pas modifiées mais déclenchent ou contraignent l'exécution de l'activité.

Une activité pour se déclencher doit avoir reçu toutes les données de contrôle et toutes les entrées correspondant aux données d'une équation d'activation de l'activité. Chacune de ces équations précise quelles sont les données nécessaires à l'exécution de l'activité, et les données produites.

Ces équations sont du type: "$E_1\ E_3\ C_1\ \overline{C_2}\ M_2 \Rightarrow S_1\ \overline{E_1}\ E_3$" signifiant que l'on a besoin des entrées E_1, E_3, du mécanisme M_2, et du contrôle C_1 pour démarrer l'activité associée, mais que l'on ne doit pas avoir le contrôle C_2. Lorsque les conditions sont réalisées, l'activité fournira la sortie S_1, mais ne "rendra" pas la donnée E_1 (qui sera dite consommée), tandis que E_3 ne sera pas modifiée.

Toute activité doit recevoir au moins une donnée de contrôle, sans quoi elle ne pourra jamais être déclenchée.

Une donnée de contrôle qui est aussi une entrée sera représentée uniquement comme un contrôle. La séparation contrôle/entrée apparaîtra dès lors au niveau de décomposition suivant.

Graphiquement, les données de contrôles sont représentées par les flèches arrivant au somment des boîtes.

Les données en sortie

Elles représentent les résultats produits en sortie par l'activité émettrice.

Toute activité doit posséder au moins une donnée de sortie. Il peut s'agir de données à destination d'autres activités du même diagramme (entrée ou contrôle), ou à destination du contexte de l'activité, c'est-à-dire que ces données sont des résultats de l'activité mère.

Les mécanismes

Les mécanismes correspondent aux ressources (matérielles, logicielles, humaines) utilisées pour le bon déroulement d'une activité.

Ils sont représenté graphiquement par les flèches reliées à la base des boîtes. Une flèche **entrant** dans la boîte indique que le mécanisme, participant à la réalisation de l'activité, est décrit dans un diagramme qui lui est associé.

Une flèche **sortant** de la boîte indique par contre que le mécanisme n'est pas décrit dans un autre diagramme.

5.2.3. Les DataGrams

Les boîtes

Dans un DataGram, les boîtes représentent les données. Chaque boîte sera identifiée par un nom.

Une description des données sera fournie sous un formalisme laissé à l'entière discrétion du concepteur.

Les flèches

Dans ce diagramme, les flèches représentent les activités. Le contenu de ces activités est défini par les ActiGrams.

Chaque flèche possède une étiquette reprenant le nom de l'activité correspondante.

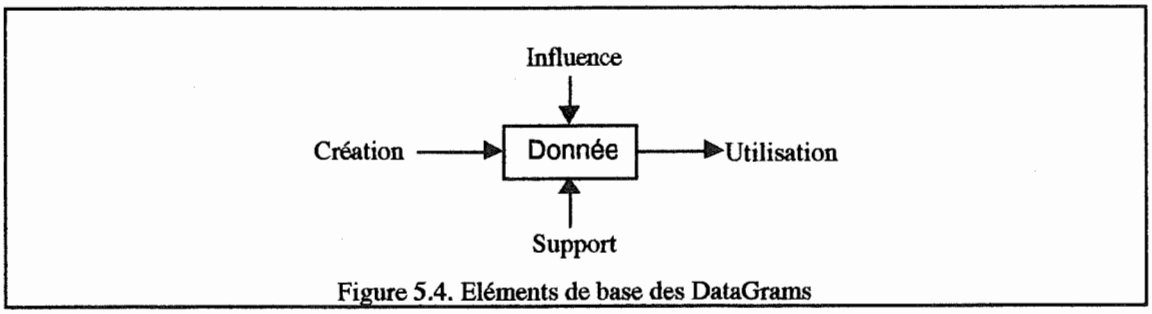


Figure 5.4. Eléments de base des DataGrams

Activité créatrice

Les données sont créées par des opérations (activités). Dans les ActiGrams ces données correspondent aux résultats produits par les activités. On représente les activités créatrices par les flèches aboutissant au côté gauche des boîtes.

Activité utilisatrice

Les données peuvent être utilisées par des activités qui les reçoivent en entrée. On représente ces activités par les flèches sortant du côté droit des boîtes.

Activité de contrôle

Les données subissent, de la part d'activités, des "influences" et des "contrôles" lors de leur création et de leur utilisation. Ces contrôles et influences correspondent à la vérification par des activités de contraintes définies sur les données.

Les supports

Tous comme les activités ont recours à des mécanismes, les données sont stockées sur des supports. Graphiquement, cela est représenté par les flèches aboutissant au bas des boîtes.

5.2.4. Conventions de représentation

Afin d'alléger les représentations graphiques, diverses conventions ont été adoptées. Ces conventions sont valables autant pour les DataGrams que pour les ActiGrams.

Les flèches peuvent se diviser ou se regrouper afin de représenter les différents usages d'un même élément, ou les utilisations conjointes d'éléments distincts. Par exemple, une donnée est utilisée comme contrôle pour plusieurs activités, ou plusieurs flux de données se regroupent pour devenir des entrées pour une activité.

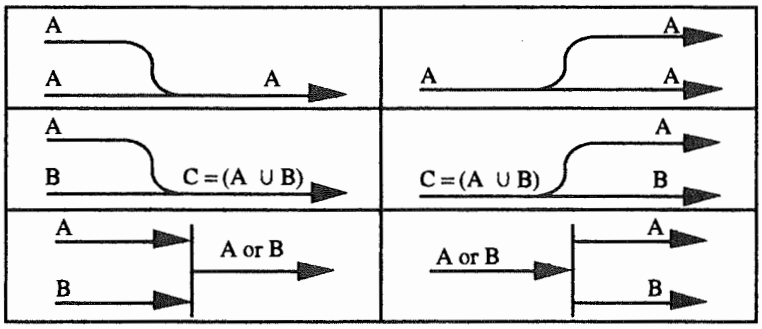
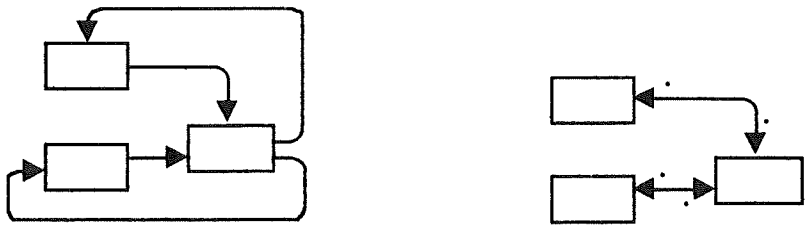


Tableau 5.1. Conventions graphiques d'unification et de division de flux.

Lorsque deux activités dialoguent, et seulement dans ce cas, on peut utiliser des flèches à double sens. Ces flèches remplacent les deux flèches qui reliaient, chacune dans un sens, les activités. On ajoute un point près des pointes de flèches pour indiquer leur double sens (figure 5.5).

L'étiquette associée à la flèche est composée des étiquettes des deux flèches séparées par une barre de division ("/"). L'ordre est important: la première étiquette est la plus importante des deux. De plus, on utilisera la flèche associée à cette étiquette pour représenter la simplification.



a. Schéma de base

b. Simplification graphique

Figure 5.5.

5.3. Démarche

La phase d'analyse d'un système produit deux ensembles de diagrammes. Le premier de ces ensembles contient les diagrammes décrivant le système en termes d'activités. Le second contient la description en termes de données.

Chacun des diagrammes subira ce qu'on appelle un "author-reader cycle". Il s'agit en fait de confronter le diagramme réalisé, par le concepteur, à un lecteur familier du problème à modéliser. Le diagramme ainsi revu sera retravaillé par son auteur.

Les diagrammes suivront ce cycle jusqu'au moment où auteurs et lecteurs seront d'accord quant à l'exactitude et la complétude des diagrammes réalisés.

Les commentaires sur les diagrammes seront obligatoirement fait par écrit afin de garder un suivi de l'évolution du modèle. Cette façon de faire, engendre de la part des auteurs et des utilisateurs un réel dialogue.

En fin de phase de conception, les ActiGrams et DataGrams seront confrontés afin de déceler d'éventuelles contradictions.

De nombreuses méthodes existent pour passer des diagrammes à une implémentation physique du système. La majorité de ces méthodes restreignent les possibilités de modélisation des diagrammes en ajoutant des contraintes sur les diagrammes obtenus en fin d'analyse. Il s'agira, après la phase d'analyse, de dériver, des différents ActiGrams produits, les traitements et leur structuration, de les confronter au hardware choisi et de les implémenter.

Partie II

Méta-modèles

Préliminaire

Pour chacune des cinq méthodes présentées dans la première partie de ce travail, nous allons maintenant élaborer un méta-modèle.

Ces cinq méta-modèles serviront de base à l'élaboration du méta-modèle global, objectif de ce travail.

Le méta-modèle global sera construit à partir du méta-modèle d'une méthode, auquel nous tenterons de faire correspondre les entités et associations des autres méta-modèles.

Il est évident que ces correspondances se feront de manière plus ou moins aisée en fonction des similitudes entre concepts des différentes méthodes. Nous serons amené à compléter le méta-modèle par de nouvelle entités et associations.

Les méta-modèles élaborés ci-après seront exprimés dans un formalisme entité/association issu des travaux de CHEN [CHE. 76]. Le formalisme utilisé est basé sur la description qu'en ont fait Y. PIGNEUR et F. BODART [Bod. 89].

Nous n'avons utilisé que les concepts standards de ce type de modèle afin de faciliter la lecture par des personnes habituées aux formalismes de MERISE.

De plus, afin d'alléger la représentation graphique et de faciliter la perception de tels modèles, nous n'avons pas représenté de manière graphique les attributs propres aux entités et aux associations, ni les dépendances entre rôles et associations. Une description plus précise de ces attributs est donnée dans la description des entités qui les possèdent.

6. Méta-modèle de la méthode MERISE

Sur base des concepts définis dans la méthode MERISE, nous allons élaborer deux méta-modèles: le premier sera basé sur les concepts "statiques" extraits des modèles des données, et le second sur les concepts extraits des modèles dynamiques.

6.1. Concepts Statiques

La figure 6.1 ci-dessous donne un aperçu du méta-modèle basé sur les concepts statiques décrits ci-après.

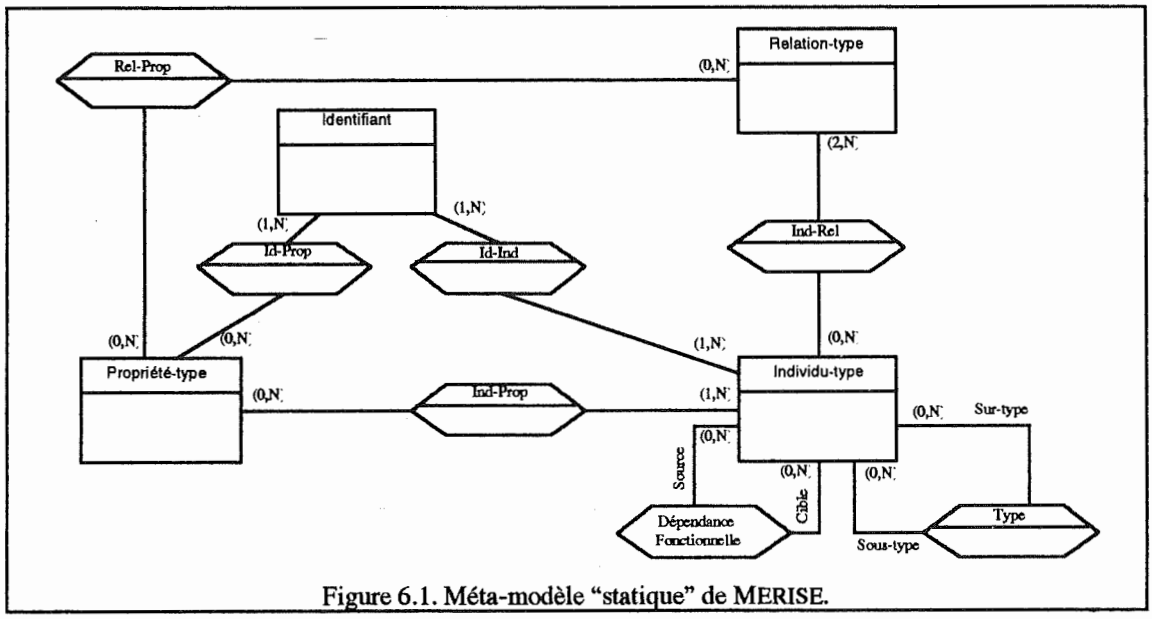


Figure 6.1. Méta-modèle "statique" de MERISE.

L'entité Individu-type

Un *Individu-type* est une classe d'individus qui ont en commun un ensemble de propriétés. Un *Individu-type* est décrit par:

- un nom, qui l'identifie
- une définition

L'entité Propriété-type

Une *Propriété-type* est une classification des propriétés semblables de toutes les occurrences d'individus appartenant à un individu-type ou de toutes les occurrences de relations appartenant à une relation-type.

Une *Propriété-type* est décrite par:

- un nom. Ce nom, concaténé au nom de l'*Individu-type* qui possède cette propriété-type (association Ind-Prop) est identifiant.

- une définition
- un domaine: définition du domaine des valeurs qui peuvent être prises par une occurrence de cette propriété.

L'association Ind-Prop

Cette association entre les entités *Individu-type* et *Propriété-type* représente la fait que les occurrences de cet *Individu-type* ont en commun un ensemble de propriétés.

- Un *Individu-type* est associé à au moins une *Propriété-type*
- Une *Propriété-type* est associée à zéro, un ou plusieurs *Individus-types*.

L'entité Identifiant

Un identifiant est une propriété, ou un groupe de propriétés, permettant de cerner de manière univoque toute occurrence de l'*Individu* identifié.

On remarquera que l'identifiant est un attribut. Nous en avons fait une entité à part entière pour permettre de représenter le fait qu'un identifiant peut être non seulement un attribut, mais également un groupe d'attributs, et qu'un même individu peut posséder plusieurs identifiants. De plus, cela permet de mettre en évidence le rôle joué par cet "attribut".

Un *identifiant* possède comme attribut:

- ID : identifiant de l'*identifiant*.
- Rang: détermine s'il s'agit d'un identifiant principal ou non (booléen).

L'association Ind-Id

Cette relation représente le fait qu'un *Identifiant* identifie un *Individu-type*

- Tout *Individu-type* est identifié par un ou plusieurs *Identifiants*.
- Tout *Identifiant* identifie un et ou plusieurs *Individus-types*.

L'association Id-Prop

Cette association met en relation un *Identifiant* et la ou les *Propriétés-types* qui le composent.

On a une dépendance fonctionnelle d'inclusion des associations Id-Prop et Id-Ind vers l'association Ind-Prop. En effet, un *Individu-type* ne peut être identifié que par un *Identifiants* composé de *Propriétés* que possède cet *Individu-type*.

- Tout *Identifiant* est constitué d'au moins une *Propriété-type*.
- Une *Propriété-type* est constituante de zéro, un ou plusieurs *Identifiants*.

L'entité Relation-type

Une *Relation-type* est une relation définie sur un ou plusieurs *Individus-types.*Une *Relation-type* est décrite par :

- son nom
- sa définition: texte décrivant la relation du monde réelle modélisée par cette *Relation-type.*
- sa dimension: le nombre d'*Individus-types* qui y participent.

L'association Ind-Rel

Association entre les entités *Individu-type* et *Relation-Type* qui modélise la participation d'un *Individu-type* à une *Relation-type.*

Cette association possède deux attributs: la cardinalité et le nom du rôle joué par l'*Individu-type.*

- Un *Individu-type* peut participer à zéro, une ou plusieurs *Relations-types*
- Une *Relation-type* participe au moins deux fois à la relation Ind-Rel.

L'association Rel-Prop

Association entre les entités *Propriété-type* et *Relation-Type* qui exprime le fait qu'une *Relation-type* peut avoir des *Propriétés-types.*

- Une *Relation-Type* est associée à zéro, une ou plusieurs *Propriétés-types*
- Une *Propriété-type* est associée à zéro, une ou plusieurs *Relations-types.*

Remarque:

Toute *Propriété-type* doit au moins être associée à un *Individu-type* ou à une *Relation-type.*

L'Association Dépendance Fonctionnelle

La *Dépendance Fonctionnelle* inter-individus traduit le fait que connaissant une occurrence (*source*) de l'un des deux individus composant la collection de la relation, on connaît directement une et une seule occurrence de l'autre individu (*cible*)".

On remarquera que les dépendances fonctionnelles sont des contraintes d'intégrité.

Une *Dépendance Fonctionnelle* est décrite par les attributs suivants:

- un nom
- une description textuelle
- un caractère faible ou fort (booléen)
- stable ou non (booléen)

L'association *Dépendance Fonctionnelle* porte sur un *Individu-type* jouant le rôle de "source" et un *Individu-type* jouant le rôle de "cible".

- Un *individu-type* participe à zéro, une ou plusieurs *Dépendances Fonctionnelles* en tant que source.
- Un *individu-type* participe à zéro, une ou plusieurs *Dépendances Fonctionnelles* en tant que cible.

L'association *Type*

Elle permet de représenter le fait qu'un *Individu-type* est une généralisation ou une spécialisation d'un autre *Individu-type* . Par exemple,l'*Individu-type* **Personne** est une généralisation des *Individus-types* **Patient** et **Médecin.**

Pour modéliser la spécialisation, il suffit de permettre à plusieurs *Individus-types* d'être des généralisations d'un même *Individu-type.*

- Un *Individu-type* peut être un sur-type (type générique) de zéro, un ou plusieurs autres *Individus-types* .
- Un *Individu-type* peut être un sous-type de zéro, un (généralisation) ou plusieurs (spécialisation) autres *Individus-types.*

6.2. Concepts dynamiques

La figure 6.2 ci-dessous présente les concepts utilisés dans Merise pour décrire la dynamique du système.

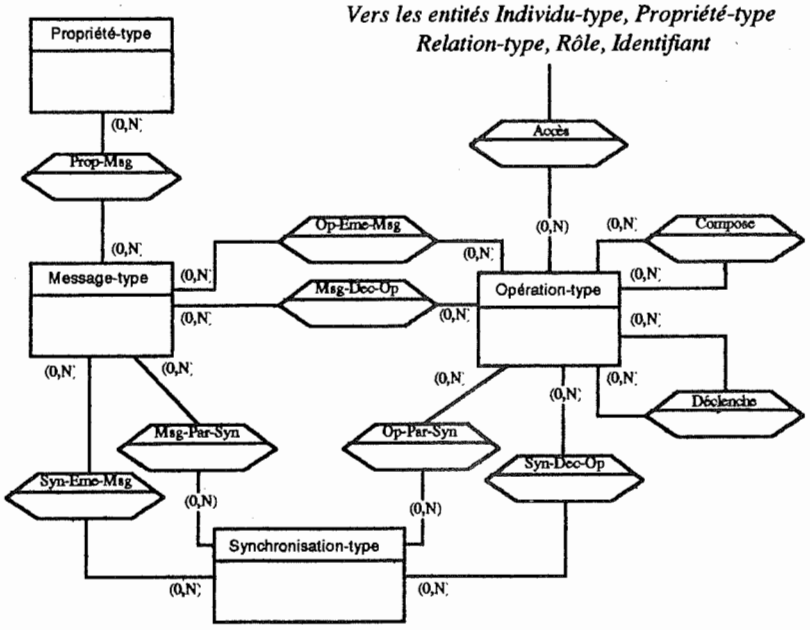


Figure 6.2. Méta-modèle "dynamique" de MERISE.

Le méta-modèle de la méthode MERISE est donc finalement représenté par l'association des figures 6.1 (Méta-modèle statique) et 6.2 (Méta-modèle dynamique).

On remarque aisément que les seuls liens entre ces deux modèles sont les associations *Prop-Msg* et les associations d'accès: *Acc-Ind, Acc-Rel, Acc-Prop, Acc-Rol, Acc-Id.*

L'entité Message-type

Un *Message-type* est un ensemble de messages possédant les mêmes caractéristiques (même source interne ou externe; même conséquences). Un *Message-type* est décrit, en plus des associations auxquelles il participe par les attributs suivants:

- son nom
- sa description sous forme textuelle (source si externe au système)
- le nombre de survenances possibles de ce message.

Remarque:

La notion d'événement n'est pas représentée par une entité particulière. Cela est dû au fait qu'un événement est soit l'occurrence d'un message-type, soit la terminaison d'une opération ou encore la réalisation d'une synchronisation.

L'association Prop-Msg

Cette association modélise le fait qu'un message peut contenir des informations correspondant à des valeurs de propriétés.

On associe donc l'entité *Message-type* à l'entité *Propriétés-types* .

- Un *Message-type* peut être associé à zéro, une ou plusieurs *Propriétés-types.*
- Une *Propriétés-types* peut être associée à zéro, un ou plusieurs *Message-type* .

L'entité Opération-type

Une *Opération-type* est une classe d'opérations qui correspondent aux mêmes ensembles d'actions à accomplir en réaction à un événement. On remarquera que dans MERISE, la distinction entre opération et exécution de cette opération (processus) n'est pas réalisée.

Les attributs de cette entité sont:

- le nom de l'*Opération-type*
- la description de la ou des actions à effectuer

• la durée (moyenne) d'exécution de l'*Opération-type.*

L'association Compose

Une *Opération-type* peut être composée d'une ou plusieurs autres *Opérations-types.* Cette relation permet une simplification des descriptions des actions à exécuter.

La cardinalité est (0:N) pour chaque rôle.

L'association Déclenche

On représente via cette association le fait qu'une *Opération-type* peut en déclencher une ou plusieurs autres. Cette association est réflexive sur l'entité *Opération-type* .

La cardinalité est (0:N) pour chaque rôle.

L'association Op-Eme-Msg

Une Opération peut émettre, en fin d'exécution, des messages. Les messages ainsi produits seront du même type. On associe donc *Opération-type* et *Message-type.*

Cette association Op-Eme-Msg possède un attribut facultatif, intitulé *condition* , qui contient l'énoncé de l'éventuelle condition d'émission du *Message-type* par l'*Opération-type.*

• Une *Opération-type* peut émettre zéro, un ou plusieurs *Message-types.*

• Un *Message-type* peut être émis par zéro (événement externe), une ou plusieurs *Opérations-types.*

L'association Msg-Dec-Op

La production d'un message peut déclencher l'exécution d'une Opération. On associe donc les *Opérations-types* et aux *Messages-types* qui les déclenchent.

• Une *Opération-type* peut être déclenchée par zéro, un ou plusieurs *Message-types.*

• Un *Message-type* peut déclencher zéro, une ou plusieurs *Opérations-types.*

L'entité Synchronisation-type

Une *Synchronisation-type* est une classe de synchronisation possédant les mêmes caractéristiques.

Les *Synchronisations-types* sont principalement décrites par les associations auxquelles elles participent; en plus des attributs suivants:

- un nom
- la description sous forme textuelle de la synchronisation
- L'énoncé de la condition de réalisation sous forme de proposition logique
- Le délai de synchronisation.

L'association Msg-Par-Syn

Cette association permet de modéliser le fait qu'un *Message-type* participe à une *Synchronisation-type*. On attribue aux occurrences de cette relation les attributs suivants:

- Une durée de participation du *Message-type* à la *Synchronisation-type*
- Un critère de sélection de l'occurrence du *Message-type* qui participera à la réalisation de la synchronisation.

Les cardinalités de cette association sont les suivantes:

- Un *Message-type* participe à zéro (événement à destination de l'environnement du système), une ou plusieurs *Synchronisations-types*.
- Une *Synchronisation-type* est associée à au moins deux événements (*Messages-types* ou terminaison d'une opération).

Remarque:

Une synchronisation en réalisation ne participe jamais directement à une nouvelle synchronisation. Il y a toujours émission d'un message.

L'association Op-Par-Syn

On représente ainsi le fait que la terminaison d'une *Opération-type* peut être un événement participant à une *Synchronisation-type*. Les cardinalités de cette association sont les suivantes:

- Une *Synchronisation-type* reçoit la participation de zéro, une ou plusieurs *Opérations-types*.
- Une *Opération-type* participe à une ou plusieurs *Synchronisations-types*.

L'association Syn-Eme-Msg

Cette association permet de modéliser le fait qu'un *Message-type* peut être émis lors de la réalisation d'une *Synchronisation-type*. Les cardinalités de cette association sont les suivantes:

- Un *Message-type* participe à zéro (événement à destination de l'environnement du système), une ou plusieurs *Synchronisations-types*.

- Une *Synchronisation-type* peut émettre zéro, un ou plusieurs *Messages-types*.

L'association Syn-Dec-Op

On représente ainsi le fait que le déclenchement d'une *Opération-type* peut être associé à la réalisation d'une *Synchronisation-type*.

- Une *Synchronisation-type* est associée à une et une seule *Opération-type*.
- Une *Opération-type* peut être associée à une ou plusieurs *Synchronisation-type*.

Remarque:

- Toute synchronisation doit au moins participer à une association Op-Par-Syn ou Msg-Par-Syn.
- Toute synchronisation-type doit au moins participer à une association Syn-Dec-Op ou Syn-Eme-Msg.

Les associations accès: Acc-Ind, Acc-Prop, Acc-Rel, Acc-Id

Ces associations représentent le fait qu'une *Opération-type* peut accéder respectivement à un individu-type, à une propriété-type, à une relation-type,et à un identifiant-type. Une *Opération-type* peut donc accéder à ces entités en vue de les consulter, créer, supprimer ou modifier.

Les associations "accès" possèdent comme attribut le type d'accès effectué: création, consultation, modification, suppression.

---

7. Méta-modèle de la méthode IDA

De chaque modèle rencontré, nous allons extraire les concepts définis et les représenter dans le méta-modèle

7.1. Modèle Entité-Association

Les concepts rencontrés dans ce premier modèle sont ceux d'entité, d'association et d'attribut.

La figure 7.1. ci-dessous présente le méta-modèle que nous avons élaboré jusqu'ici. Ce méta-modèle reprend tous les aspects statiques liés au modèle Entité-association.

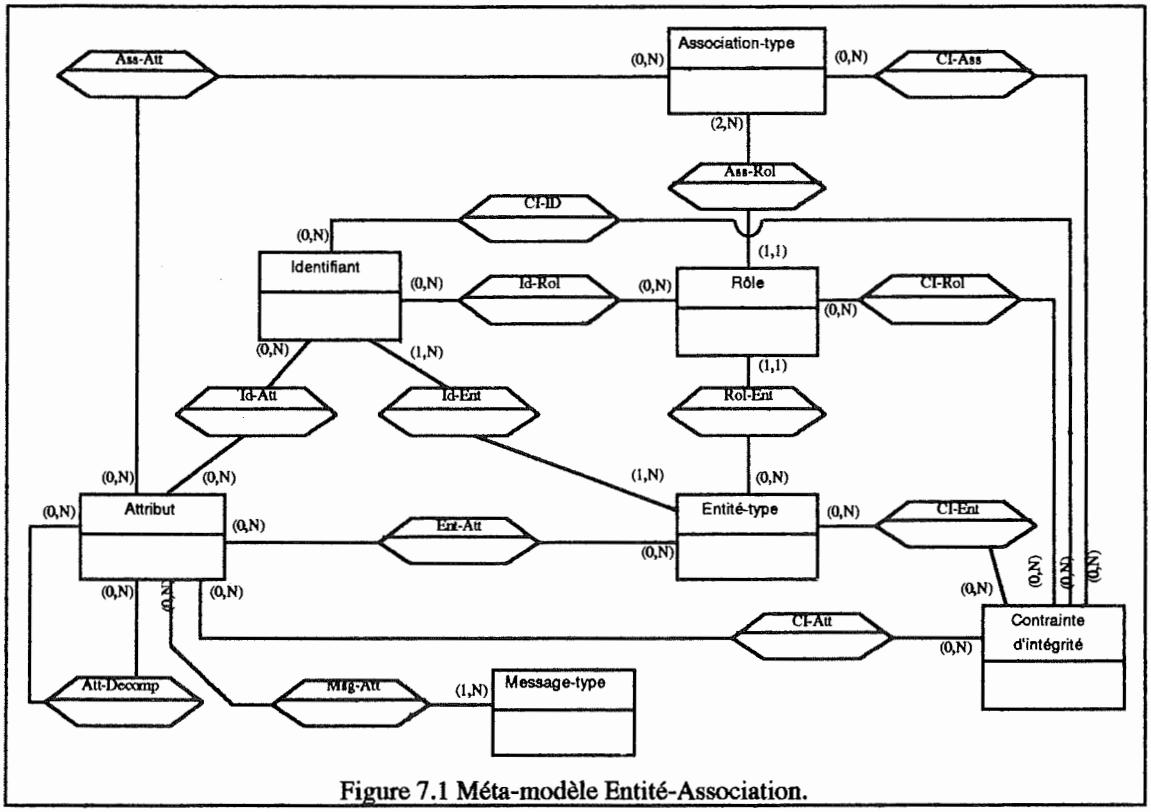


Figure 7.1 Méta-modèle Entité-Association.

L'entité *Entité-type*

Une *Entité-type* est une classe de toutes les entités possibles du réel perçu qui vérifient la définition constitutive du type.

Une *Entité-type* est décrite par:

- un nom qui l'identifie.
- une définition constitutive.
- une durée de vie ou l'énoncé d'une condition d'existence.

L'entité *Association-type*

Une *Association-type* est une classe de toute les associations possibles du réel perçu qui vérifient la définition constitutive du type.

Une *Association-type* possède les attributs suivants:

- un nom
- une définition constitutive
- une durée de vie ou l'énoncé d'une condition d'existence.

L'entité Rôle

Cette entité correspond à un rôle joué par une *Entité-type* dans une *Association-type* . Elle est décrite par les attributs suivants: le nom et la connectivité du rôle. Nous y ajouterons le caractère répétitif indiquant si plusieurs entités d'un même type peuvent jouer simultanément le même rôle (multivalué).

L'association Ass-Rol

L'association Ass-Rol met en relation chaque *Association-type* et les *Rôles* qu'elle fait jouer aux entités.

- Toute *Association-type* possède aux moins deux *Rôles* .
- Tout Rôle est associé à une et une seule *Association-type* .

L'association Rol-Ent

Cette association représente le fait qu'un *Rôle* est joué par une *Entité-type.*

- Tout *Rôle* est joué par une et une seule *Entité-type.*
- Une *Entité-type* peut jouer zéro, un ou plusieurs *Rôles.*

L'entité Attribut

Un *Attribut* est une caractéristique ou une qualité d'une entité ou d'une association. Il peut prendre un(e) ou plusieurs valeurs ou groupes de valeurs.

Un *Attribut* sera donc décrit par les associations auxquelles il participe et les attributs suivants:

- un nom
- une description de ce qu'il représente dans le monde réel
- une booléen représentant le caractère répétitif ou simple de l'*Attribut.*
- un booléen représentant le caractère facultatif ou obligatoire de l'*Attribut.*
- un domaine de valeur (facultatif si l'attribut est décomposable)

L'association Att-Décomp

Cette association entre entités *Attributs* représente le fait qu'un attribut peut se décomposer en plusieurs autres *Attributs*.

- Un *Attribut* se décompose en zéro (élémentaire), un ou plusieurs autres *Attributs*.
- Un Attribut fait partie de la décomposition de zéro, un ou plusieurs *Attributs* .

L'association Ent-Att

On exprime par cette association le fait qu'une *Entité-type* peut posséder divers *Attributs*.

- Une *Entité-type* est associée à zéro (rarissime), un ou plusieurs *Attributs*.
- Un *Attribut* peut être associé à zéro, une, ou plusieurs *Entités-types*.

L'association Ass-Att

Comme les *Entités* , une *Association* peut posséder zéro, un ou plusieurs *Attributs* , qui peuvent être associés à zéro, une ou plusieurs *Associations*.

Remarque:

- Tout *Attribut* doit au moins participer à une occurrences d'une des trois associations suivantes: *Ass-Att* , *Ent-Att* ou *Att-Décomp*.

L'entité Identifiant

Cette entité permet d'exprimer le fait qu'un *Attribut* , un groupe d'*Attributs* , ou les *Rôles* qu'elle joue, identifie de manière unique toute occurrence d'un *Individu-type*.

En plus d'un nom (qui l'identifie), cette entité possède un attribut indiquant s'il s'agit de l'identifiant principal de l'*Entité-type* associée.

L'association Id-Ent

Cette association entre les entités *Entité-type* et *Identifiant* associe au moins un *Identifiant* à chaque *Entité-type*.

- Tout *Identifiant* est associé à au moins une *Entité-type*.
- Toute *Entité-type* est associée à au moins un *Identifiant*. Sauf dans les rares cas où l'entité-type ne possède aucun attribut.

L'association Id-Att

Un *Identifiant* peut être composé d'*Attributs* non-facultatifs.

- Un *Identifiant* peut être composé de zéro, un ou plusieurs *Attributs.*
- Un *Attribut* peut composer zéro, un, ou plusieurs *Identifiants.*

L'association Id-Rôle

Un *Identifiant* peut être basé sur les *Rôles* que joue une *Entité-type.*

- Un *Identifiant* peut être composé de zéro, un ou plusieurs *Rôles.*
- Un *Rôle* peut composer zéro, un, ou plusieurs *Identifiants.*

Remarques:

- Un *Identifiant* peut être composé d'*Attributs* et de *Rôles.*
- Tout *Identifiant* doit au moins participer à une association Id-Att ou Id-Rôle.

L'entité CI (Contrainte d'intégrité)

Chaque occurrence de cette entité correspond à une contrainte d'intégrité facultative (Existence, Inclusion, Exclusion et Egalité de rôle, et d'association).

On a déjà représenté les contraintes d'intégrité obligatoires (connectivité et identification).

L'entité *CI* possède les attributs suivants:

- un nom
- l'énoncé de la contrainte
- la description du contexte dans lequel elle doit être vérifiée.
- le type de contrainte (existence, inclusion, exclusion, égalité de rôles et d'associations, valeur, dépendance fonctionnelle

Associations CI-Ent, CI-Ass, CI-Att, CI-Rol, CI-Id

Nous allons associer l'entité *CI* aux éléments sur lesquels elle peut porter (associations CI-Ent, CI-Ass, CI-Att, CI-Rol, CI-Id). On pourrait également l'associer aux traitements qui doivent la respecter.

Les rôles joués par l'entité CI dans ces associations, ainsi que ceux joués par les différents éléments (*Entité-type*, *Attributs*, *Association-type*, *Rôles* et *Identifiant*), ont tous la connectivité (0:N).

On peut donner à ces associations un attribut indiquant le rôle joué par l'élément associé à une contrainte d'intégrité (source ou cible).

L'entité Message-type

Cette entité correspond au concept de message. Un *Message-type* est défini par un nom, une description, et par les associations Msg-mémoire auxquelles il participe. La description d'un message externe (émis ou reçu par aucun traitement: Cf. infra) sera complétée par un échéancier de survenance.

L'association Msg-Att

Cette association représente le fait qu'un message contient des informations qui appartiennent à la mémoire du système. Un message-type contient au moins une information, ne fut-ce que l'heure de sa survenance.

7.2. Modèle de structuration des traitements

Le schéma ci-dessous reprend les concepts relatifs aux traitements définis dans les modèles de structuration, de la statique et de la dynamique des traitements.

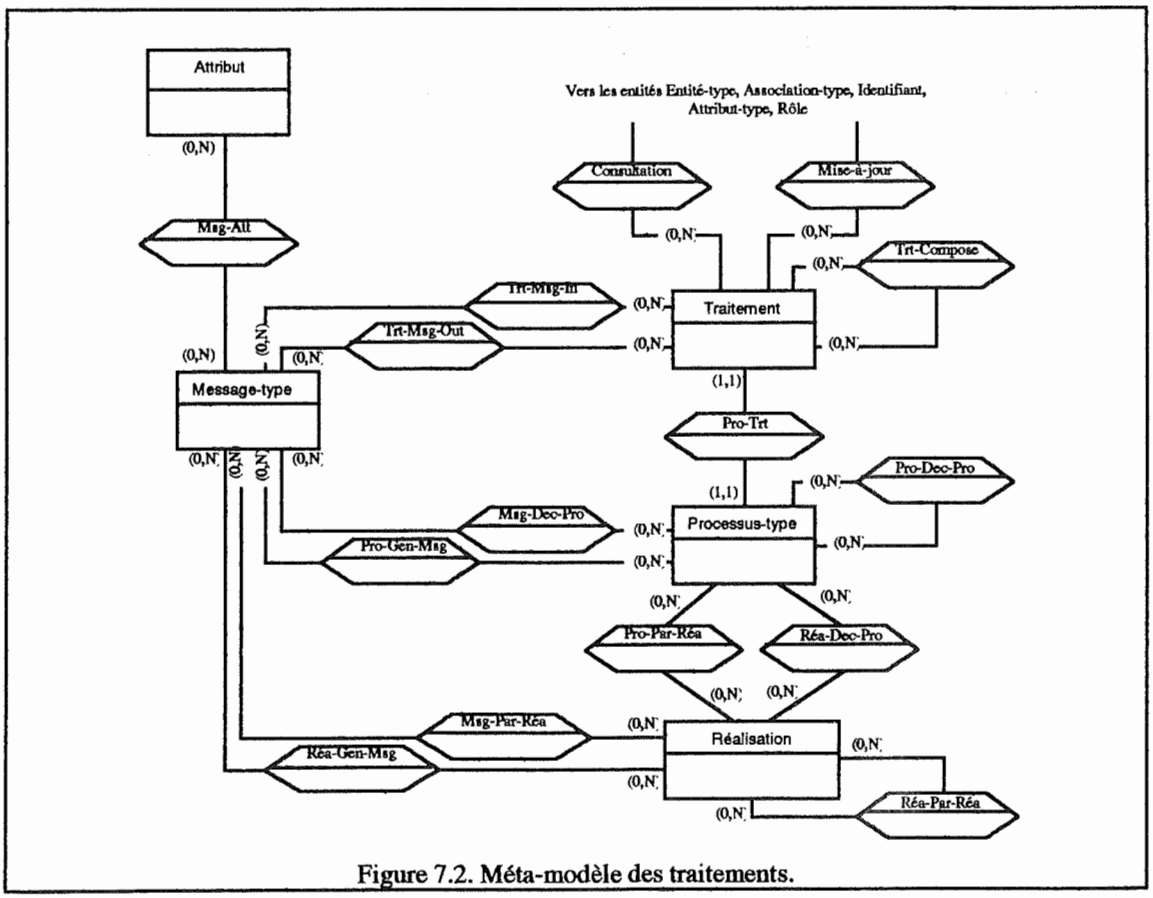


Figure 7.2. Méta-modèle des traitements.

L'entité Traitement

Chaque occurrence de cette entité *Traitement* correspond à un traitement. On décrit un *Traitement* par les propriétés suivantes:

- un nom
- une définition
- un objectif
- les performances fonctionnelles
- un niveau hiérarchique (projet, application, phase, fonction)
- l'énoncé des règles d'actions du traitement: cet attribut n'est obligatoire que pour les traitements de niveau fonction.
- le caractère automatisé, manuel ou interactif du traitement.

L'association Trt-Compose

Cette association permet de représenter la structuration hiérarchique des traitements.

On associera tous les *Traitements* , de niveau supérieur au niveau fonction, aux *Traitements* de niveau directement inférieur qui le composent. De même tous les *Traitements* , sauf les projets, seront associés au *Traitement* de niveau directement supérieur qu'ils composent.

- Un *Traitement* de niveau projet se décompose en au moins un *Traitement* de niveau application, et ne compose aucun autre *Traitement* .
- Un *Traitement* de niveau application compose un seul *Traitement* de niveau projet, et se décompose en au moins un *Traitement* de niveau phase.
- Un *Traitement* de niveau phase compose un seul *Traitement* de niveau application, et se décompose en au moins un *Traitement* de niveau fonction.
- Un *Traitement* de niveau fonction compose un seul *Traitement* de niveau phase, et ne se décompose en aucun autre *Traitement* .

7.3. Modèle de la statique des traitements

L'association Trt-Msg-In

Cette association relie les *Messages-types* aux *Traitements* qui les reçoivent en entrée.

- Un *Message-type* peut être reçu par zéro (message externe), un ou plusieurs *Traitements* .
- Un *Traitement* reçoit zéro, un ou plusieurs *Messages-types* .

L'association Trt-Msg-Out

Cette association relie les *Traitements* aux *Messages-types* qu'ils génèrent en sortie. On attribue à cette association la condition d'émission par le *Traitement* du *Message* associé.

- Un *Message-type* peut être émis par zéro (message externe), un ou plusieurs *Traitements* .
- Un *Traitement* émet zéro, un ou plusieurs *Messages-types* .

Les associations de consultations

Ces associations sont les suivantes: *Cslt-Ent, Cslt-Ass, Cslt-Att, Cslt-Id,* et *Cslt-Rôle.*

Elles permettent de représenter, respectivement, la consultation des *Entités-types* , des *Associations-types* , des *Rôles* , des *Attributs* , et des *Identifiants* , comme informations reçues en entrée par un *Traitement.*

- Un *Traitement* reçoit zéro, une ou plusieurs informations (*Entité-type* , *Association-type* , *Rôle* , *Attribut* , *Identifiant*) en entrée.
- Une information peut être reçue par zéro, un ou plusieurs *Traitements.*

Les associations de mise à jour

Il s'agit des associations *Maj-Ent, Maj-Ass, Maj-Att, Maj-Id,* et *Maj-Rôle.*

Ces associations représentent respectivement la mise à jour des *Entités-types,* des *Associations-types* , des *Rôles* , des *Attributs* , et des *Identifiants* , en tant qu'informations produites en sortie par un *Traitement.*

Chacune de ces associations possède un attribut déterminant le type de mise à jour: création, modification, suppression.

- Un *Traitement* met à jour zéro, une ou plusieurs informations (*Entité-type* , *Association-type* , *Rôle* , *Attribut* , *Identifiant*).
- Une information peut être mise à jour par zéro, un ou plusieurs *Traitements.*

Remarques:

- Tout *Traitement* doit au moins participer à une association de consultation de la mémoire ou Trt-Msg-In, ainsi qu'à une association de mise à jour de la mémoire ou Trt-Msg-Out.
- Tout Traitement devra respecter les contraintes d'intégrité définies sur les informations qu'il met à jour.

- Afin de simplifier la représentation graphique du méta-modèle, les associations Cslt-Ent, Cslt-Ass, Cslt-Att, Cslt-Id, Cslt-Rôle, Maj-ent, Maj-ass, Maj-att, Maj-id, et Maj-rôle sont représentées par les associations "Consultation" et "Mise-à-jour".

7.4. Modèle de la dynamique des traitements

L'entité Processus-type

A chaque Traitement correspond un type de processus. Ce processus-type est destiné à représenter l'exécution d'un traitement. On remarquera que IDA, contrairement à MERISE, propose une telle distinction.

- Nom identifiant le processus
- Durée estimée d'exécution

L'association Pro-Trt

Elle fait correspondre à un Traitement le type de processus correspondant.

- Un *Processus-type* correspond à un et un seul *Traitement*
- Un *Traitement* peut être associé à zéro ou un *Processus-type*.

L'association Msg-Dec-Pro

Cette association représente le fait qu'un message, en génération, peut déclencher l'exécution d'un traitement, c'est-à-dire le *Processus-type* associé au traitement.

- Un *Message* peut déclencher zéro, un ou plusieurs *Processus-types*.
- Un *Processus-type* peut être déclenché par zéro, un ou plusieurs *Messages*.

Remarque:

- Comme pour Merise, les événements ne constituent pas une entité particulière. Un événement est soit l'occurrence d'un message-type, soit la terminaison d'un processus ou encore la réalisation d'une synchronisation.

L'association Pro-Dec-Pro

On représente ainsi le fait qu'un *Processus-type* , en terminaison, peut en déclenche un autre.

- Un *Processus-type* peut déclencher zéro, un ou plusieurs *Processus-types*.
- Un *Processus-type* peut être déclenché par zéro, un ou plusieurs *Processus-types*.

L'association Pro-Gén-Msg

Cette association représente le fait qu'un *Message* est généré par un *Processus-type* .

- Un *Message* peut être généré par zéro, un ou plusieurs *Processus-types.*
- Un *Processus-type* peut générer zéro, un ou plusieurs *Messages.*

Synchronisation, Duplication & Sélection

Nous allons regrouper les notions de Duplication, Sélection et Synchronisation en une seule entité: l'entité *Réalisation.* En effet, ces trois concepts diffèrent peu. Ils correspondent tous trois à l'attente d'événements stimulants, qui devront vérifier une condition (triviale pour les duplications) afin de créer en réaction de nouveaux événements.

L'entité Réalisation

Une *Réalisation* possède comme attribut:

- un nom
- un type: synchronisation, duplication ou sélection
- une condition de réalisation. Pour les duplications, on précisera leur expression
- un nombre qui correspond au facteur de duplication, ou le nombre d'occurrences de l'unique événement contributif des synchronisations qui n'en possède qu'un.

Remarques:

- Toute réalisation doit au moins avoir un événement contributif, ce qui revient à participer à une des associations suivantes: Msg-Par-Syn, Pro-Par-Syn, ou Syn-Par-Syn (Cf. infra).
- Si une synchronisation n'a qu'un événement contributif, alors il faut préciser dans la condition de réalisation le nombre d'occurrences nécessaires de cet événement à la réalisation de cette synchronisation.

L'association Réa-Gen-Msg

Une *Réalisation,* en réalisation, peut générer zéro, un ou plusieurs *Messages.* De plus, un *Message* peut être généré par zéro, une ou plusieurs *Réalisations.*

L'association Réa-Dec-Pro

Une *Réalisation*, lorsque sa condition est vérifiée, peut déclencher zéro, un ou plusieurs *Processus* . Un *Processus-type* peut être déclenché par zéro, une ou plusieurs *Réalisations.*

L'association Pro-Par-Réa

Un *Processus-type* en terminaison peut participer à zéro, une ou plusieurs *Réalisations* . Une *Réalisation* peut recevoir la participation d'un nombre quelconque de *Processus-types.*

L'association Msg-Par-Réa

A l'instar d'un *Processus-type* , un *Message* peut également participer à des *Réalisations.*

- Un *Message* participe à zéro, une ou plusieurs *Réalisations.*
- Une *Réalisation* reçoit la participation de zéro, un, ou plusieurs *Messages.*

L'association Réa-Par-Réa.

Elle représente le fait que la réalisation d'une *Réalisation* peut participer à la réalisation d'une autre *Réalisation* . Aucune restriction n'est faite sur le nombre de participations.

7.5. Modèle des ressources

La figure 7.3 ci-dessous donne un aperçu des concepts que l'on a rencontrés dans le modèle des ressources et le diagramme des flux.

Entité ressource

"Une ressource représente un certain volume de moyens physiques identiques dont l'organisation doit disposer".

En plus de son nom, et de sa description, nous allons attribuer à cette entité sa capacité et son calendrier de disponibilité.

Association réquisition

Cette association entre les entités Traitement et Ressource exprime le fait qu'un Traitement nécessite certaines ressources pour pouvoir s'exécuter.

L'association Réquisition possède l'attribut taux_de_réquisition qui indique la quantité requise par le traitement de cette ressource.

- Toute *Ressource* est requise par au moins un *Traitement.*
- Un *Traitement* peut requérir zéro, une ou plusieurs *Ressources* pour s'exécuter.

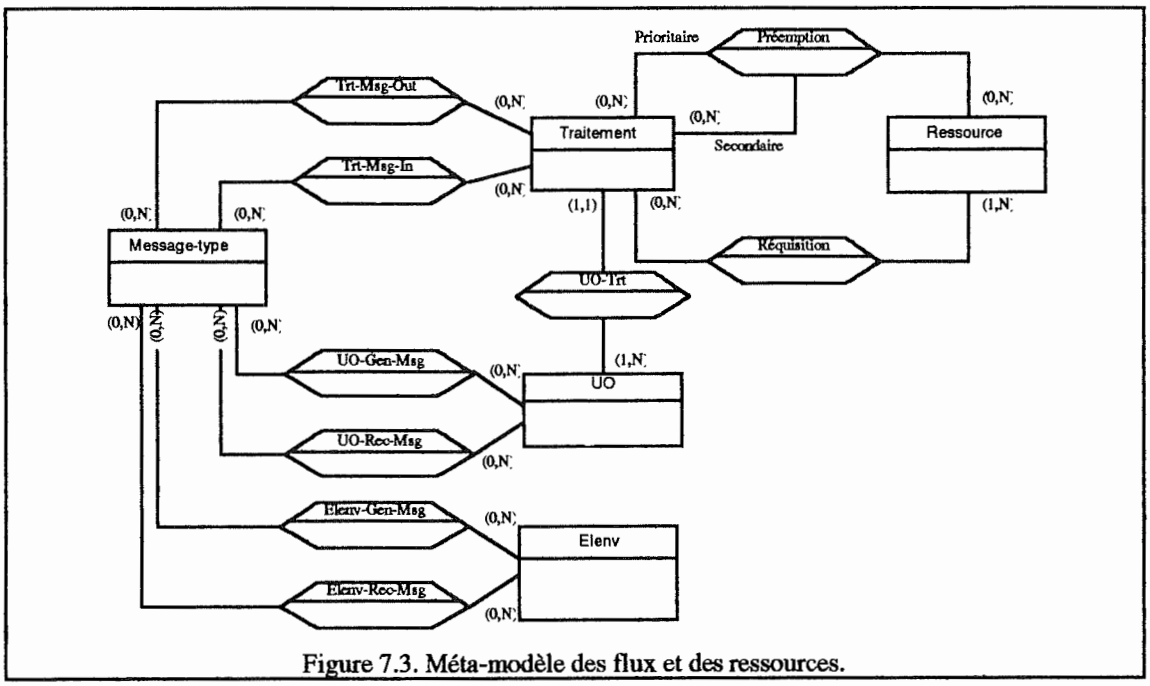


Figure 7.3. Méta-modèle des flux et des ressources.

Association préemption

Cette association représente le droit de préemption que possède un *Traitement* (prioritaire) sur un autre (secondaire) pour l'utilisation d'une *Ressource.* Cette association est ternaire.

- Un *Traitement* peut jouer le rôle "prioritaire" dans zéro, une ou plusieurs associations *préemption.*
- Un *Traitement* peut jouer le rôle "secondaire" dans zéro, une ou plusieurs associations *préemption.*
- Une *Ressource* peut participer à zéro, une ou plusieurs associations *préemption.*

7.6. Diagramme des flux

Entité UO

Cette entité représente les unités organisationnelles de l'organisation. Chaque unité organisationnelle sera décrite par un nom.

Entité Elenv

Les occurrences de cette entité correspondent aux éléments de l'environnement du S.I. L'entité *Elenv* possède comme attribut un nom.

L'association UO-Trt

Toute unité organisationnelle est associée aux traitements dont elle est responsable.

- Toute UO est responsable d'au moins un Traitement.
- Tout Traitement est géré par une et une seule UO.

Les associations UO-Gen-Msg & Elenv-Gen-Msg

Les associations UO-Gen-Msg et Elenv-Gen-Msg relient les messages respectivement aux unités organisationnelles ou aux éléments de l'environnement qui les génèrent.

- Un *Message* est généré par zéro, une ou plusieurs *UO* ou *Elenv*.
- Une UO, ou un *Elenv* génère zéro, un, ou plusieurs *Messages*.

Les associations UO-Rec-Msg & Elenv-Rec-Msg

Les associations UO-Rec-Msg et Elenv-Rec-Msg relient les messages respectivement aux unités organisationnelles ou aux éléments de l'environnement qui les reçoivent.

- Un *Message* est généré par zéro, une ou plusieurs *UO* ou *Elenv*.
- Une UO, ou un *Elenv* génère zéro, un, ou plusieurs *Messages*.

Remarque:

- Toute UO, respectivement tout Elenv, doit participer au moins à une association UO-Rec-Msg ou UO-Gen-Msg, respectivement Elenv-Rec-Msg ou Elenv-Gen-Msg.

---

8. Méta-modèle de Rémora

Pour élaborer le méta-modèle de la méthode REMORA, nous allons nous limiter aux modèles descriptif et conceptuel.

Le modèle logique ne sera pas pris en compte, puisqu'il aborde les problèmes d'implémentation.

8.1. Modèle Descriptif

Ce modèle est basé sur trois concepts de base: les objets, les opérations et les événements.

La figure 8.1 ci-dessous représente le méta-modèle élaboré sur base des éléments du modèle descriptif.

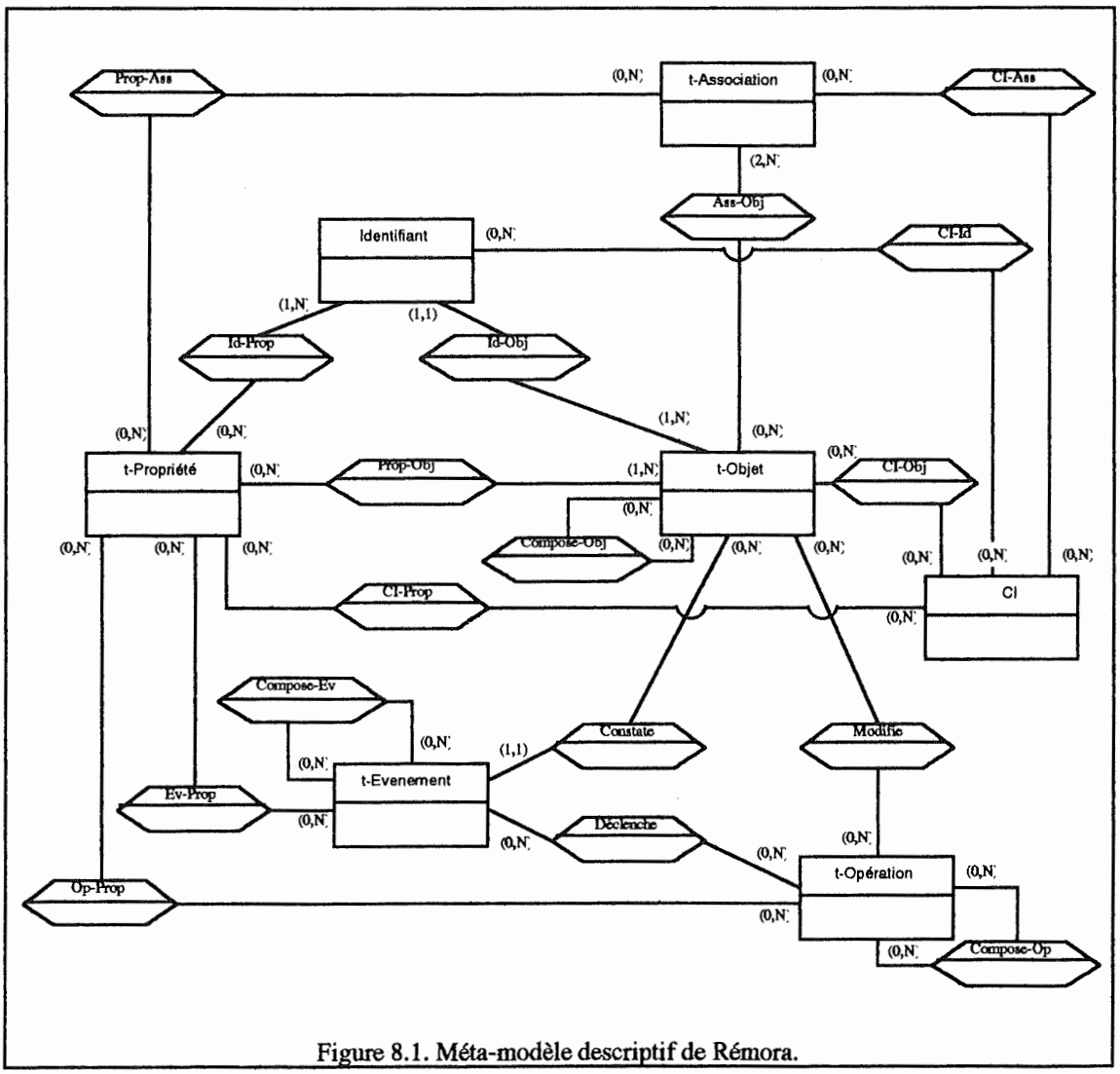


Figure 8.1. Méta-modèle descriptif de Rémora.

8.1.1. Les objets

L'entité *t-Objet*

Un *t-Objet* est une classe d'objets définis par des propriétés sémantiquement identiques. Un *t-Objet* possède comme attributs:

- un nom. Ce nom est identifiant.
- une définition

L'entité *t-Propriété*

Une *t-Propriété* est "l'abstraction d'une seule caractéristique possédée par toutes les entités, elles-mêmes considérées comme un *objet* ".

L'entité *t-Propriété* possèdent les attributs suivants:

- un nom. Ce nom, concaténé au nom de l'*Objet* associé par la relation *Prop-Obj* est identifiant de la *t-Propriété* .
- une définition.
- un domaine de valeurs: il s'agit de la définition de l'ensemble des valeurs que peut prendre la *t-Propriété* .

L'entité *t-Association*

Une *t-Association* est l'abstraction d'un ensemble d'associations qui ont lieu systématiquement entre différentes choses du monde réel.

Une *t-Association* possède les attributs suivants:

- un nom, permettant de l'identifier.
- une description.

L'association Prop-Obj

Cette association met en relation les entités *t-Objet* et *t-Propriété* . Elle représente le fait qu'un *t-Objet* possède des propriétés.

- Tout *t-Objet* possède au moins une *t-Propriété* .
- Une *t-Propriété* peut être possédée par zéro, un ou plusieurs *t-Objets*.

L'association Prop-Ass

Cette association, reliant les entités *t-Association* et *t-Propriété* , représente le fait qu'une *t-Association* peut posséder des *t-Propriétés* .

- Une *t-Association* possède zéro, un ou plusieurs *t-Propriétés* .
- Une *t-Propriété* peut être possédé par zéro, une ou plusieurs *t-Associations*.

L'association Ass-Obj

Reliant les entités *t-Objet* et *t-Association* , cette association concrétise le fait que des *t-Objets* sont mis en rapport par des *t-Associations* .

- Toute *t-Association* met en rapport au moins deux *t-Objets*.
- Un *t-Objet* peut être associé à zéro, une ou plusieurs *t-Associations* .

On attribue à la relation Ass-Obj un attribut précisant le rôle joué par le *t-Objet* dans la *t-Association* , et un attribut donnant la connectivité de ce rôle.

L'entité *Identifiant*

Un identifiant est un attribut ou un groupe d'attributs permettant de cerner de manière univoque toute occurrence du *t-Objet* identifié.

Nous en avons fait une entité à part entière pour permettre de représenter le fait qu'un identifiant peut être une propriété seule, un groupe de propriétés, et qu'un même objet peut posséder plusieurs identifiants.

Un *Identifiant* possède comme attribut:

- ID : identifiant de l'*identifiant*.
- Rang: détermine s'il s'agit d'un identifiant principal ou non (booléen).

L'association Id-Obj

Cette relation représente le fait qu'un *Identifiant* identifie un *t-Objet*.

- Tout *t-Objet* est identifié un ou plusieurs *Identifiants*.
- Tout *Identifiant* identifie un et un seul *t-Objet*.

L'association Id-Prop

Cette association met en relation un *Identifiant* et le ou les *t-Propriétés* qui le composent.

On a une dépendance fonctionnelle d'inclusion de les associations Id-Prop et Id-Obj vers l'association Prop-Obj. En effet, un *t-Objet* ne peut être identifié que par un *Identifiant* composé de *t-Propriétés* que possède ce *t-Objet*.

Association Compose-Obj

Cette association permet de représenter le fait qu'un *t-Objet* peut être composé d'autres *t-Objets*.

- Un *t-Objet* est composé de zéro, un ou plusieurs autres *t-Objets*.
- Un *t-Objet* peut être composante de zéro ou un autre *t-Objet*.

Entité CI & Associations CI-Obj, CI-Ass, CI-Prop, CI-ID

Cette entité et ces associations permettent de représenter le fait qu'une contrainte d'intégrité (CI) peut être définie sur les divers éléments du modèle entité/association.

Zéro, une ou plusieurs contraintes d'intégrité peuvent être définies sur chaque entité.

8.1.2. Les opérations

L'entité t-Opération

Une *t-Opération* est un ensemble d'opérations qui "modifient des objets ou des associations d'objets appartenant respectivement aux mêmes classes et définies sur des propriétés sémantiquement identiques".

Chaque *t-Opération* est décrite par diverses propriétés:

- un nom
- la description (nature) de l'opération type.

L'association Op-Prop

Cette association exprime le fait qu'une opération type peut posséder des propriétés la situant dans l'espace et le temps. Il s'agit notamment des propriétés décrivant les règles d'action de l'opération.

L'association Modifie

Cette association est définie entre les entités *t-Opération* et *t-Objet*. Elle représente la classe d'opération Modifie définie entre une *t-Opération* et un *t-Objet*.

On pourra préciser le type de modification dont il s'agit. Il faut néanmoins remarquer que l'on ne peut pas créer ou supprimer un objet: on peut seulement le faire passer dans l'état existant ou inexistant.

- Un *t-Objet* peut être modifié par zéro, une ou plusieurs *t-Opérations.*
- Une *t-Opération* peut modifier zéro, un ou plusieurs *t-Objets.*

L'association Compose-Op

Cette association représente le fait qu'une opération peut être composée de zéro, une ou plusieurs autres *t-Opérations.*

8.1.3. Les événements

L'entité t-Evénement

Un t-Evénement représente une classe d'événements constatant des changements d'états de même nature, et déclenchant des opérations d'une même classe.

Un t-Evénement est décrit par son nom et les associations auxquelles il participe.

L'association Ev-Prop

On représente ainsi le fait qu'un événement peut posséder diverses propriétés, notamment l'heure de sa survenance.

L'association Constate

Cette association relie les t-Evénements aux t-Objets. Elle représente la classe d'associations MODIFIE définies entre événements et objets.

- Tout *t-Evénement* constate un *t-Objet.*
- Un *t-Objet* peut être "constaté" par zéro, un ou plusieurs *t-Evénements.*

L'association Déclenche

Cette association relie les entités t-Evénements et t-Opérations. Elle représente la classe d'associations DECLENCHE définies entre événements et opérations. Cette association possède un attribut représentant l'éventuelle condition de déclenchement de l'opération par l'événement.

- Tout *t-Evénement* peut déclencher zéro, une ou plusieurs *t-Opérations.*
- Une *t-Opération* peut être déclenchée par zéro, un ou plusieurs *t-Evénements.*

L'association Compose-Ev

Cette association représente le fait qu'un *t-Evénement* peut être composé de zéro, une ou plusieurs autres *t-Evénements.*

8.2. Modèle Conceptuel

La figure 8.2 ci-dessous montre le méta-modèle basé sur les notions conceptuelles définies dans REMORA.

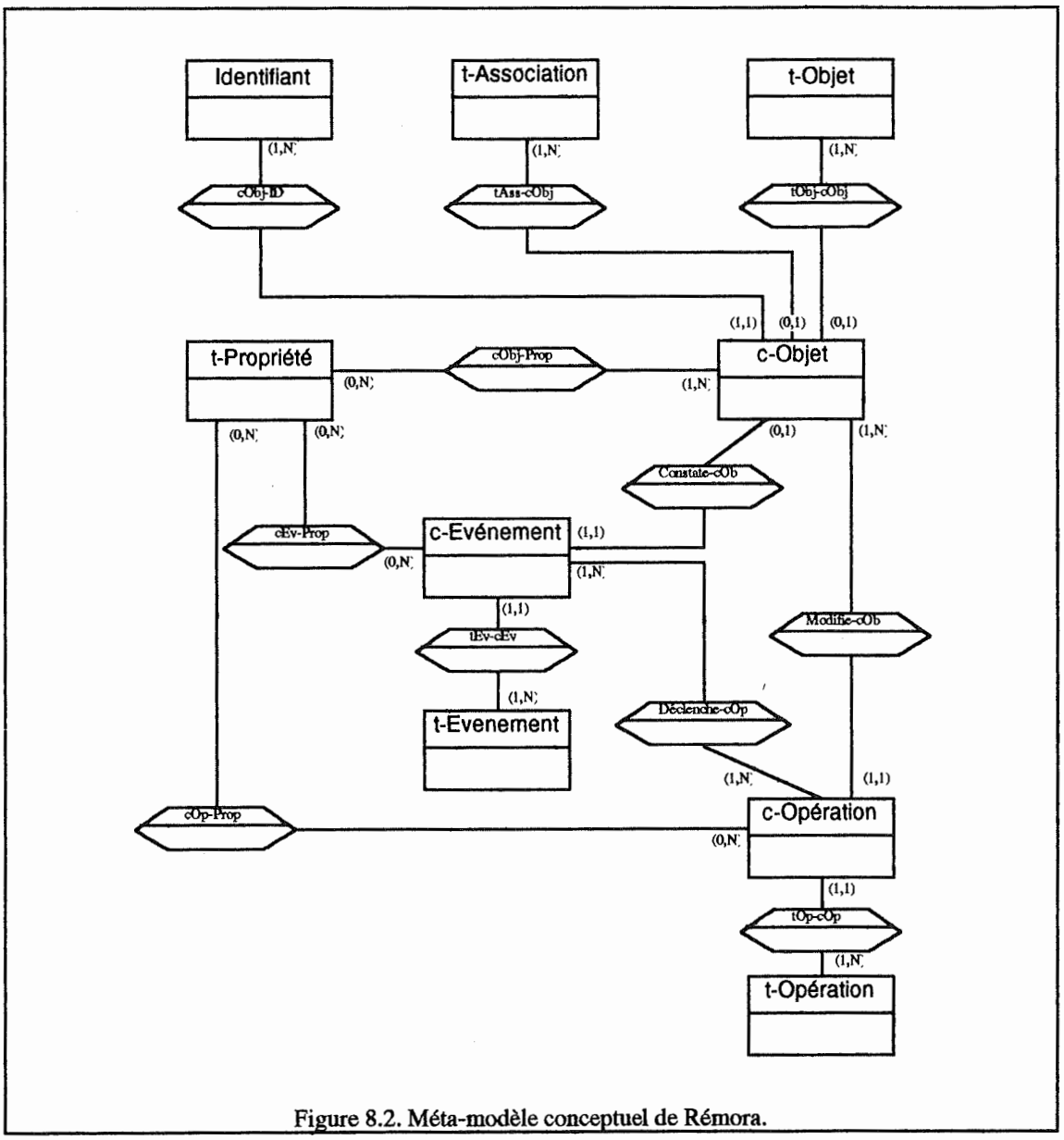


Figure 8.2. Méta-modèle conceptuel de Rémora.

8.2.1. Représentation des c-Objets

L'entité c-Objet

Elle représente la normalisation des *t-Objets* et *t-Associations*. On lui attribue les propriétés suivantes:

- un nom
- un type: RP (relation permanente) ou RV (relation variable).

L'association cObj-Prop

Cette association associe des propriétés non décomposables aux *c-Objets*, en vue de représenter la relation. Tout *c-Objet* possède au moins une propriété.

L'association cObj-ID

Elle permet de représenter qu'une propriété est l'identifiant d'un *c-Objet*. Tout *c-Objet* possède un identifiant. De plus, elle permet de représenter le fait qu'une association sera représentée par un *c-Objet* par une relation (au sens relationnel) possédant comme attribut les identifiants des entités y participant.

L'association tObj-cObj

Un *t-Objet*, une fois normalisé, correspond à un ou plusieurs *c-Objets*. Cette association permet de le représenter. Tous les *c-Objets* associés à un même *t-Objet* forment une c-classe.

L'association tAss-cObj

Une *t-Association* sera "transformée" en un *c-Objet* , dont les attributs seront les identifiants des *t-Objets* participant à cette association.

8.2.2. Représentation des c-opérations

L'entité c-Opération

Une *c-Opération* correspond à un type d'action élémentaire, exécutée en réponse à un *c-Evénement* , et modifiant un *c-Objet* . On lui attribue un nom et un type (ROP, RTEXT, REXEC).

L'association tOp-cOp

Plusieurs *c-Opérations*, de type différents, peuvent correspondre à une même t-Opération. En effet, une *t-Opération* peut avoir des aspects permanents et d'autres variables.

L'association Modifie-cOb

Une *c-Opération* peut être associée à un et un seul *c-Objet* sur lequel portera l'action correspondant à la *c-Opération.*

Tout *c-Objet* est modifié par au moins une *c-Opération.*

L'association cOp-Prop

Une *c-Opération* peut être associée à des propriétés.

8.2.3. Représentation des c-événements

L'entité c-Evénement

Cette entité correspond à un type de changement d'état élémentaire du *c-Objet* associé. On attribue, en plus d'un nom, un type à cette entité (REV, RPRED, RCONDFAC, REVARR, REVDECL).

L'association tEv-cEv

On met en relation chaque *t-Evénement* et le ou les *c-Evénements* qui permettent de le représenter sous forme normalisée.

L'association cEv-Prop

Cette association permet de mettre en relation les *c-Evénements* et les *Propriétés* qu'ils possèdent.

L'association Constate-cOb

On représente ainsi l'association d'un *c-Evénement* avec le seul *c-Objet* dont elle constate un changement d'état.

L'association Déclenche-cOp

Tout *c-Evénement* peut déclencher une ou plusieurs *c-Opérations*. Cette association possède en attribut les éventuels facteur et condition de déclenchement.

---

9. Méta-modèle de la méthode OOA

Comme nous l'avons vu, la méthode OOA comporte trois modèles: l'*Information Model* , le modèle d'états (*State Model*), et le modèle des traitements (*Process Model* ou *Action Diagram*).

Nous allons élaborer un méta-modèle sur base de ces trois modèles.

La figure ci-dessous montre le méta-modèle de la méthode OOA que nous obtenons.

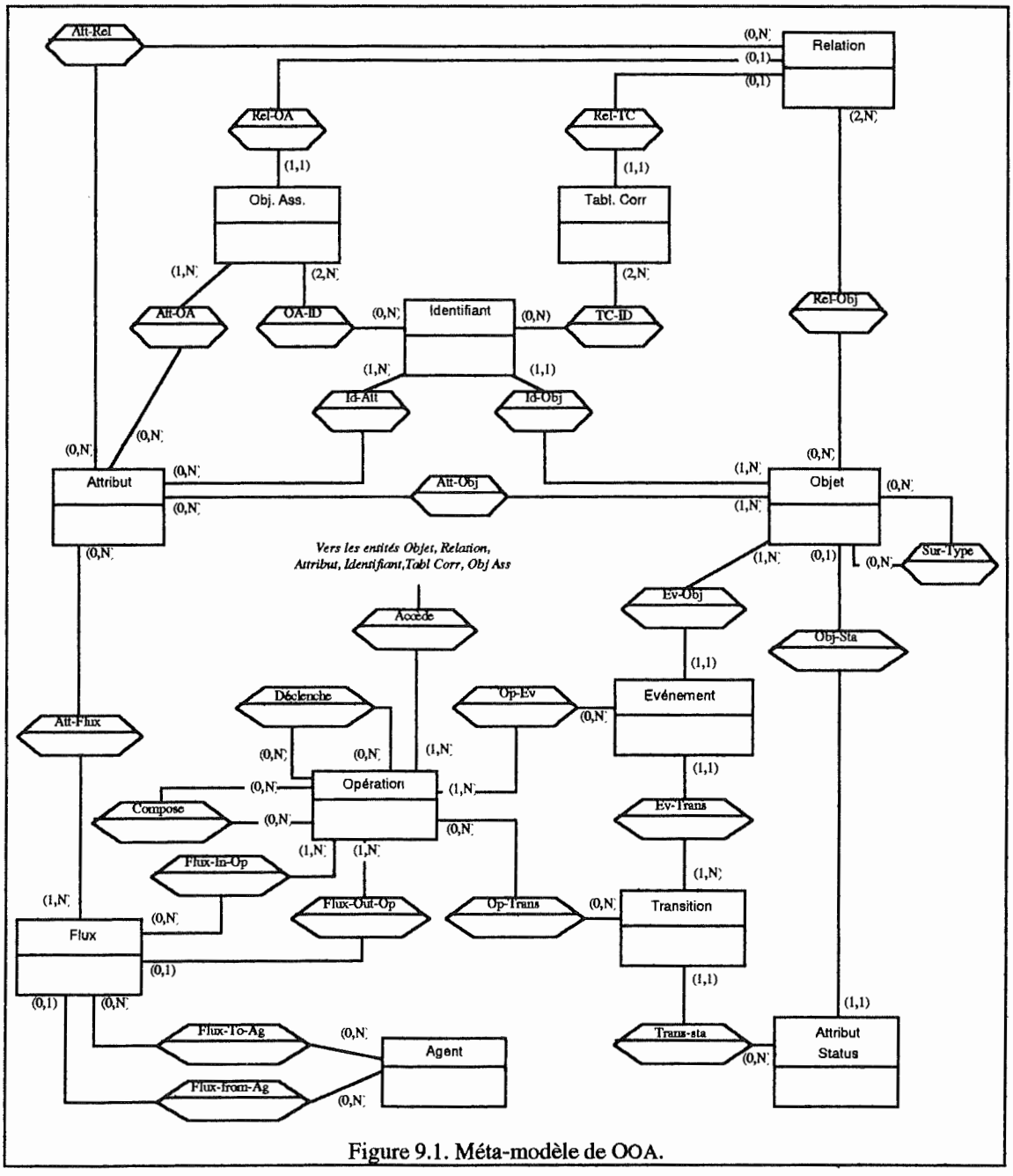


Figure 9.1. Méta-modèle de OOA.

9.1. L'information Model

Dans l'*Information Model* , nous trouvons différents concepts de base qui sont les objets, les relations et les attributs. Chacun de ces trois concepts va être considéré comme une entité.

L'entité *Objet*

Un *objet* est "une abstraction d'un ensemble de choses du monde réel telle que toutes les choses de cet ensemble possèdent les mêmes caractéristiques, respectent et sont régies par les mêmes règles".

Un *objet* possède comme attributs:

- un **nom** (chaîne de caractères). Ce nom est identifiant.
- une **définition** (texte libre)

L'entité *Attribut*

Un *Attribut* est "l'abstraction d'une seule caractéristique possédée par toutes les entités, elles-mêmes considérées comme un *objet* ".

L'entité *Attribut* possède les attributs suivants:

- un nom. Ce nom, concaténé au nom de l'*Objet* associé par la relation *Att-Obj* est identifiant de l'*Attribut.*
- une définition: reprenant notamment la référence à la relation modélisée si l'attribut est référentiel.
- un domaine de valeurs: il s'agit de la définition de l'ensemble des valeurs que peut prendre l'*Attribut.*
- un type. Cet attribut peut prendre une des valeurs suivantes: *descriptif* , *nominatif*, ou *référentiel.*

L'entité *Relation*

Une *Relation* est l'abstraction d'un ensemble d'associations qui ont lieu systématiquement entre différentes choses du monde réel.

Une *Relation* possède les attributs suivants:

- un nom, permettant de l'identifier.
- une description.
- une multiplicité, prenant une des valeurs suivantes: (1:1), (1:M), (M:M), (1:1c), (1c,1c), (1:Mc), (1c:M), (1c,Mc), (M:Mc), (Mc:Mc).

• un degré: indiquant le nombre d'*Objets* mis en relation.

L'association Att-Obj

Cette association met en relation les entités *Objet* et *Attribut* . Elle représente le fait qu'un *Objet* possède des attributs.

• Tout *Objet* possède au moins un *Attribut* , et peut en posséder plusieurs.

• Un *Attribut* peut être possédé par zéro, un ou plusieurs *Objets.*

L'association Att-Rel

Cette association, reliant les entités *Relation* et *Attribut* , représente le fait qu'une *Relation* peut posséder des *Attributs* .

• Une Relation possède zéro, un ou plusieurs Attributs.

• Un *Attribut* peut être possédé par zéro, une ou plusieurs *Relations.*

L'association Rel-Obj

Reliant les entités *Objet* et *Relation* , cette association concrétise le fait que des *Objets* sont mis en rapport par des *Relations* .

• Toute *Relation* met en rapport au moins deux *Objets.*

• Un *Objet* peut être associé à zéro, une ou plusieurs *Relations* .

On attribue à la relation Rel-Obj un attribut précisant le rôle joué par l'*Objet* dans la *Relation.*

L'entité *Identifiant*

Un identifiant est un attribut ou un groupe d'attributs permettant de cerner de manière univoque toute occurrence de l'*Objet* identifié.

Nous en avons fait une entité à part entière pour permettre de représenter le fait qu'un identifiant peut être un attribut seul, un groupe d'attributs, et qu'un même objet peut posséder plusieurs identifiants.

Un *identifiant* possède comme attribut:

• ID : identifiant de l'*identifiant.*

• Rang: détermine s'il s'agit d'un identifiant principal ou non (booléen).

L'association Id-Obj

Cette relation représente le fait qu'un *Identifiant* identifie un *Objet.*

• Tout *Objet* est identifié par un ou plusieurs *Identifiants.*

• Tout *Identifiant* identifie un et un seul *Objet*.

L'association Id-Att

Cette association met en relation un *Identifiant* et le ou les *Attributs* qui le composent.

On a une dépendance fonctionnelle d'inclusion des associations Id-Att et Id-Obj vers l'association Att-Obj. En effet, un *Objet* ne peut être identifié que par un *Identifiant* composé d'*Attributs* que possède cet *Objet*.

L'entité *Table de corrélation* (*Tab. Corr.*)

Une *Table de corrélation* est une table contenant pour chaque occurrence de la *Relation* associée, les identifiants des *Objets* reliés par cette occurrence de la *Relation*.

Elle possède un attribut permettant de l'identifier.

Exemple: la *Relation* **Emploie** associe l'*Objet* **Ouvrier** à l'*Objet* **Société**. La *Table de Corrélation* **TC1**, modélisant la *Relation* **Emploie** est composée de l'identifiant (principal) des *Objets* **Ouvrier** et **Société**.

Association Rel-TC

L'Association Rel-TC représente le fait qu'une *Table de Corrélation* est utilisée pour modéliser une *Relation*, de multiplicité (M:M) ou conditionnelle, ne possédant pas d'Attribut.

Chaque *Table de Corrélation* modélise une et une seule *Relation*.

Une *Relation* est modélisée par zéro ou une *Table de Corrélation*.

Association Tc-Id

On représente par cette association le fait qu'une *Table de Corrélation* est constituée des identifiants des *Objets* mis en rapport par la *Relation* modélisée.

On déduit la dépendance fonctionnelle suivante: Toute Table de Corrélation doit contenir les identifiants des Objets mis en rapport par la Relation qu'elle modélise.

Ce qui peut s'écrire : Soient Rel-Obj (R, O_1,O_2,...,O_n) et Id-Obj(Id_1,O_1), Id-Obj(Id_2,O_2), ..., Id-Obj(Id_n,O_n),

Si Rel-Tc (R,T) alors Tc-Id(T,Id_1), Tc-Id(T,Id_2), ... et Tc-Id(T,Id_n).

L'entité *Objet Associatif*

Un *Objet Associatif* est semblable à une *Table de Corrélation* mais est destiné à la modélisation de *Relations* possédant des *Attributs* .

De plus, on peut associer plusieurs Objets Associatifs à une même relation.

L'association Rel-OA

Elle associe les *Relations* et les *Objets Associatifs* qui les modélisent.

- Une *Relation* est modélisée par zéro ou un *Objet Associatif.*
- Un *Objet Associatif* modélise une et une seule *Relation.*

L'association Att-OA

Cette association met en relation les *Objets Associatifs* et leurs *Attributs.*

- Un *Objet Associatif* possède au moins un *Attribut.*
- Un *Attribut* peut être possédé par zéro, un ou plusieurs *Objets Associatifs.*

La dépendance fonctionnelle suivante est évidente: tout *Objet Associatif* possède les *Attributs* possédés par la Relation qu'il modélise.

Association OA-Id

On représente par cette association le fait qu'un *Objet Associatif* est constitué des identifiants des *Objets* mis en rapport par la *Relation* modélisée.

On a la dépendance fonctionnelle suivante: tout *Objet Associatif* doit contenir les identifiants des Objets mis en rapport par la Relation qu'il modélise.

Association Sur-Type

Cette association permet de représenter le fait qu'un Objet peut être sur-type d'un autre Objet.

- Un Objet est sur-type de zéro, un ou plusieurs autres Objets
- Un Objet peut être sous-type de zéro ou un autre Objet.

9.2. Modèle d'états

Etat, transition, événement et opération sont les principaux concepts définis dans ce modèle.

L'entité Attribut Status

Au cours de sa vie, un *Objet* peut se trouver dans divers états. L'attribut status, ajouté à chaque *Objet*, permet de connaître l'état dans lequel se trouve l'*Objet*.

A cet effet, nous créons une nouvelle entité *Attribut Status* .Cette entité a une existence autonome de l'entité *Attribut* afin de mieux représenter ses caractéristiques et les rôles qu'elle joue.

L'attribut principal de l'entité *Attribut Status* est la définition de son domaine de valeurs. Ce domaine définit l'ensemble des états dans lesquels pourra se trouver l'*Objet* associé.

Association Obj-Sta

Cette association modélise le fait qu'un objet possède divers états, et possède donc un attribut status.

Un *Objet* est associé au maximum à un *Attribut Status*, et tout *Attribut Status* est associé à un et un seul *Objet*.

Entité Transition

Une *Transition* est le passage d'un *Objet* d'un état à un autre. Une transition est donc définie par un état de départ, et un état d'arrivée. Ces deux états prenant leur valeur dans le domaine de définition de l'objet associé à l'*Attribut Status* cible de la transition (Cf. infra).

On a donc comme attribut de cette entité:

- un identifiant de la transition.
- un état de départ, et un état d'arrivée.

Remarque

- Nous aurions pu représenter les états d'arrivée et de départ en associant les transitions avec un nouvelle entité "Etat". Les occurrences de cette entité correspondraient aux valeurs que peut prendre l'attribut status.

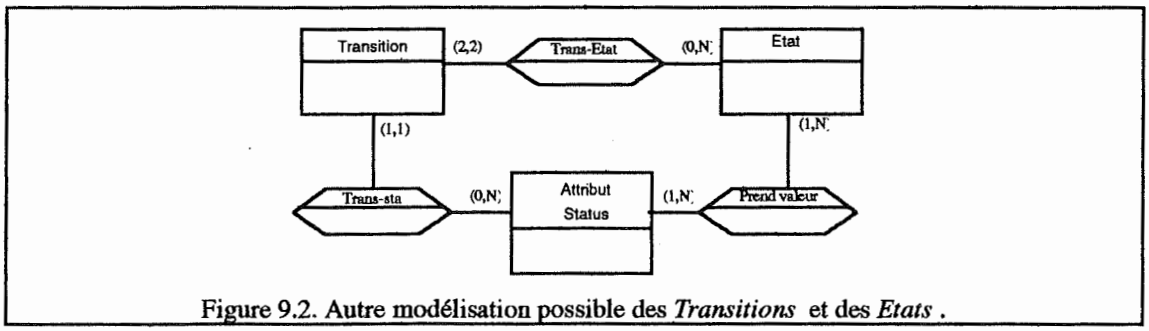


Figure 9.2. Autre modélisation possible des *Transitions* et des *Etats* .

• Nous n'avons pas choisi cette représentation car elle risque de perturber la bonne compréhension du phénomène. Les états ne sont pas des concepts à part entière mais bien des valeurs d'attribut.

Association Trans-Sta

Cette association représente le fait qu'une *Transition* porte sur la modification de l'état d'un *Objet* , par la modification de son *Attribut Status.*

Une *Transition* ne porte que sur un et un seul *Attribut Status* , mais un même *Attribut Status* peut être la cible de plusieurs *Transitions.*

Entité Evénement

Un *Evénement* représente la survenance d'une modification dans le système ou dans son environnement.

Un *Evénement* est soit d'origine interne au système (généré par une *Opération*), soit externe. La destination d'un *Evénement* est toujours un *Objet* . De plus, une transition est toujours associée à un *Evénement* .

Un *Evénement* est décrit par :

• un nom l'identifiant

• une description de l'*Evénement*

Association Ev-Obj

Cette association représente le fait qu'un *Evénement* est destiné à un *Objet* . Charge à cet *Objet* de "réagir".

• Tout *Evénement* est associé à un et un seul *Objet.*

• Un *Objet* peut être la destination de plusieurs *Evénements.*

Association Ev-Trans

Par cette association, on représente le fait qu'un *Evénement* peut provoquer une transition d'état de *l'Objet* auquel est destiné l'*Evénement* .

• Tout *Evénement* est associé à une *Transition* .

• Toute *Transition* est associée à un ou plusieurs *Evénements* .

Entité Opération

Une Opération est une suite d'instructions exécutables par le système de manière manuelle, partiellement ou totalement automatique.

Une *Opération* est définie par son nom et sa description sous une forme quelconque (pseudo-code, langage naturel, ...), et par les associations auxquelles elle participe.

Association Op-Trans

Une *Opération* peut être déclenchée par une *Transition* . Quand on passe dans un état particulier, on déclenche une suite d'*Opérations*.

Cette association possède un attribut ("# ordre") qui donne le numéro d'ordre d'exécution de l'*Opération* associée.

- Toute *Transition* est associée à une ou plusieurs *Opérations*
- Une *Opération* peut être déclenchée par zéro, une ou plusieurs *Transitions*.

Association Op-Ev

Cette association permet de représenter le fait qu'une *Opération* peut générer un ou plusieurs *Evénements* à destination d'*Objet*.

- Une *Opération* génère un ou plusieurs *Evénements*
- Un *Evénement* peut être généré par zéro (Evénement externe), une ou plusieurs *Opérations*.

9.3. Modèle des traitements

L'entité Flux

Cette entité va nous permettre de représenter les flux de données circulant entre les Opérations.

Un *Flux* est décrit par:

- Son nom
- Sa description

De plus, on associera au *Flux* les *Attributs* correspondant aux informations qu'il transporte, ainsi que les éventuelles *Opérations* source et destination du *Flux*.

L'association Flux-Out-Op

Cette association entre les entités *Opération* et *Flux* représente le fait qu'une *Opération* est la source d'un *Flux*.

- Un *Flux* est originaire d'au plus une *Opération*
- Une *Opération* Peut être la source de zéro, un ou plusieurs *Flux*.

L'association Flux-In-Op

Cette association entre les entités *Opération* et *Flux* représente le fait qu'une *Opération* est la destination d'un *Flux.*

- Un *Flux* est destiné à au plus une *Opération*
- Une *Opération* peut être la destination de zéro, un ou plusieurs *Flux.*

L'entité Agent

Cette entité représente les agents internes et externes intervenants. On leur attribue un nom, une description et un type (interne ou externe).

L'association Flux-From-Ag

Cette association entre les entités *Agent* et *Flux* représente le fait qu'un *Agent* est la source d'un *Flux.*

- Un *Flux* est originaire de zéro ou un *Agent*
- Un *Agent* Peut être la source de zéro, un ou plusieurs *Flux.*

L'association Flux-To-Ag

Cette association entre les entités *Agent* et *Flux* représente le fait qu'un *Agent* est la destination d'un *Flux.*

- Un *Flux* est destiné à au plus un *Agent*
- Un *Agent* peut être la destination de zéro, un ou plusieurs *Flux.*

L'association Att-Flux

Cette association entre les entités *Attributs* et *Flux* représente le fait qu'un *Flux* peut contenir des informations correspondant aux *Attributs.*

- Un *Flux* contient zéro, un ou plusieurs *Attributs.*
- Un *Attribut* peut être contenu dans zéro, un ou plusieurs *Flux.*

L'association Accède

Cette association entre les entités *Attributs* et *Opération* représente le fait qu'une *Opération* peut accéder aux *Attributs.* en vue de les consulter, les modifier.

Ce type d'association existe aussi entre les Opération et: les Relations, les Objets, les Objets Associatifs et les Tables de Corrélation.

L'association Compose

Cette association met en relation l'entité *Opération* avec elle-même. Elle permet de représenter le fait qu'une *Opération* peut être composée de sous-Opérations.

- Une Opération est composée de zéro, une ou plusieurs autres Opérations
- Une Opération compose zéro, une ou plusieurs autres Opérations.

L'association Déclenche

Cette association met en relation l'entité *Opération* avec elle-même. Elle permet de représenter le fait qu'une *Opération* peut en déclencher une autre.

- Une *Opération* peut déclencher zéro, une autre ou plusieurs autres *Opérations*
- Une *Opération* peut être déclenchée par zéro, une autre ou plusieurs autres *Opérations*.

---

10. Méta-modèle de la méthode SADT

A partir des concepts représenter dans les différents diagrammes de SADT, nous obtenons le méta-modèle suivant:

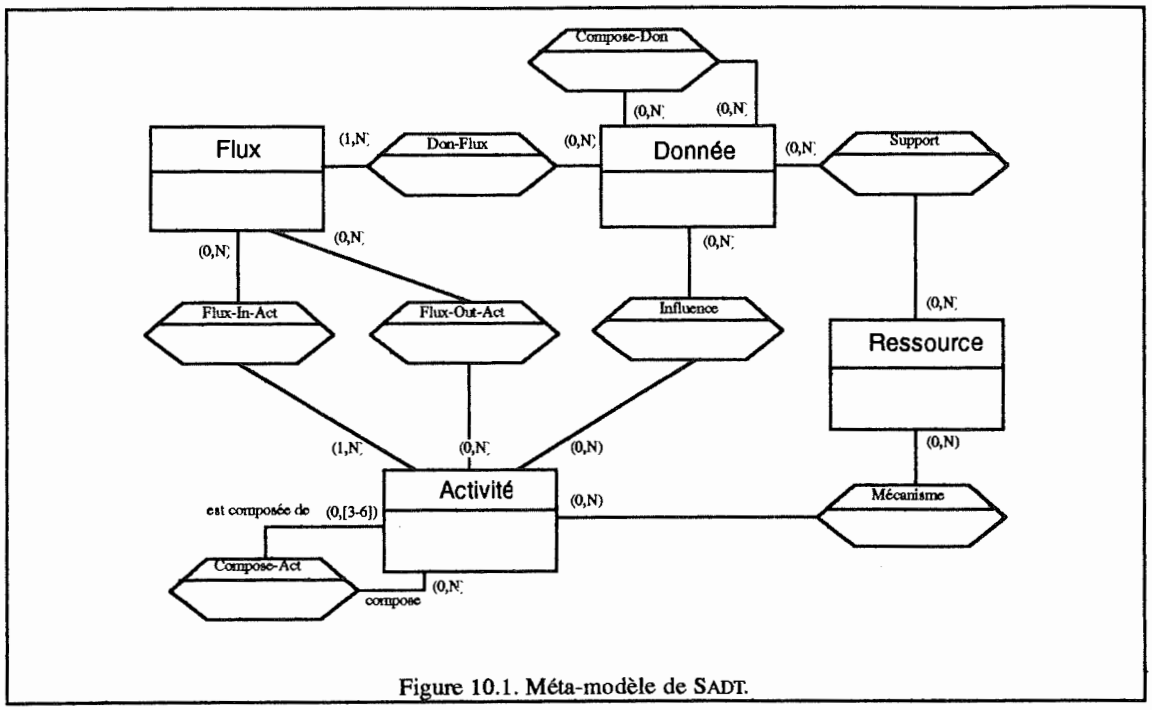


Figure 10.1. Méta-modèle de SADT.

L'entité Activité

Cette entité correspond aux activités. Pour chaque activité, on donnera son nom (qui en l'occurrence est un verbe), sa description (texte).

L'association Compose-Act

Des activités peuvent faire partie du diagramme descriptif d'une autre activité. Cette dernière est donc composée des premières, et c'est ce que représente cette association.

- Toute *Activité* peut être composée de zéro, ou de trois à six autres *Activités*.
- Une *Activité* peut être composante de zéro ou une autre *Activité*.

L'entité Donnée

Une donnée est toute information manipulable par le système. Outre son nom et sa description textuelle, aucune autre information n'est fournie. On remarquera cependant que dans la majorité des cas d'utilisation de SADT, un modèle de description des données est joint à la méthode [LIS. 90].

L'association Compose-Don

Elle représente le fait qu'une donnée peut être composée d'autres données.

L'entité Flux

Cette entité représente les flux de données entre activités. Un flux est décrit par son nom, et les associations auxquelles il participe. On remarquera que les simplifications apportées dans les représentations graphiques se limitent à ces représentations.

L'association Don-Flux

Cette association représente le fait qu'un flux de données contient diverses données.

- Un *Flux* contient au moins une *Donnée.*
- Une *Donnée* est contenue dans zéro, un ou plusieurs *Flux.*

L'association Flux-In-Act

Elle représente le fait qu'un *Flux* est destiné à une activité. Ce *Flux* pourra être une entrée ou un contrôle à destination de l'activité. Cette dernière l'utilisera ou le recevra comme un contrôle. Pour le spécifier, nous allons attribuer à l'association *Flux-In-Act* l'attribut *Type* qui peut prendre les valeurs "Contrôle" ou "Entrée".

- Une *Activité* reçoit au moins un *Flux* . Si elle ne reçoit qu'un seul *Flux,* alors il devra être du type *Contrôle.*
- un *Flux* est destiné à zéro, une ou plusieurs *Activités.*

L'association Flux-Out-Act

Elle représente le fait qu'un *Flux* est produit en sortie par une activité. Ce *Flux* correspond à une sortie ou à une création de donnée.

- Une *Activité* produit au moins un *Flux* .
- un *Flux* est produit par zéro, une ou plusieurs *Activités.*

L'association Influence

Une activité peut influencer les opérations de création et de consommation d'une donnée. Cela permet notamment de représenter des activités destinées à la vérification de contraintes d'intégrité.

- Une activité peut influencer zéro, une ou plusieurs données.
- Une données peut être influencée par zéro, une ou plusieurs activités.

L'entité Ressource

Une ressource représente un certain volume de moyens matériels, humains, ou logiciel nécessité par des activités pour leur exécution. Nous allons étendre cette définition en considérant que les supports nécessaires aux données font partie des ressources.

Pour chaque ressource, on précisera donc son nom, sa description, son type (mécanisme ou support), et sa disponibilité.

L'association Mécanisme

Elle met en rapport une ressource et les activités qui en ont besoin pour s'exécuter.

- Toute ressource (qui n'est pas un support, Cf. infra) est associée à au moins une activité.
- Une activité peut utiliser zéro, une, ou plusieurs ressources

L'association Support

Elle associe une donnée au support (qui est une ressource particulière) sur lequel elle se trouve.

- Toute donnée peut être associée à un support.
- Tout support doit être utilisé par une ou plusieurs données.

---

11. Méta-modèle global

11.1. Modèle de base

Pour créer notre méta-modèle global, reprenant les concepts définis dans les méthodes étudiées, nous allons partir du méta-modèle contenant le plus de concepts et y ajouter les concepts qu'apporterons les autres méthodes.

En ce qui concerne les données, nous allons partir du modèle entité/association décrit dans IDA. Le modèle des données proposé dans REMORA est plus complet, puisqu'on y retrouve en plus les notions de relations (au sens relationnel de CODD). Cependant, en ce qui concerne les traitements, IDA fournit le plus de concepts.

Nous choisirons donc comme méta-modèle de départ, celui réalisé pour la méthode IDA.

11.1.1. Modification du méta-modèle de IDA

Suppression de l'entité *Processus*

L'entité *Processus* correspond à la représentation de l'exécution d'un traitement. A tout *Traitement* est associé un *Processus.*

Afin de simplifier le méta-modèle global, nous allons supprimer cette entité, et attribuer ses attributs à l'entité Traitement. Un Traitement aura donc, en plus de ses anciens attributs, un attribut contenant la durée estimée d'exécution.

Outre la simplification du méta-modèle, cette suppression va nous permettre d'établir plus aisément les correspondances entre les traitements. Si nous avions conservé l'entité *Processus* , nous aurions eu une prolifération de nouvelles associations destinées à séparer les notions statiques et dynamiques non-disjointes dans la plupart des méthodes.

On considérera un *Traitement* sous un angle dynamique ou statique en fonction des relations envisagées. Lorsque l'on parlera de déclenchement de traitements, de participation à des réalisations d'un traitement en terminaison, il est évident qu'il s'agit de la notion dynamique du traitement. Lorsqu'on parlera des ressources requises, des compositions entre traitements, il s'agira de la vision statique des traitements.

Les associations Trt-Dec-Trt, Trt-Gen-Msg, Msg-Dec-Trt, Trt-Par-Réa, et Réa-Dec-Trt

Ayant supprimé l'entité *Processus*, il faut maintenant faire jouer ses rôles par l'entité *Traitement*.

L'association *Trt-Dec-Trt*

L'association *Pro-Dec-Pro*, qui représentait le déclenchement d'un processus lors de la terminaison d'un autre va être remplacée par l'association *Trt-Dec-Trt* qui représente le même phénomène entre *Traitements*. Elle conserve les connectivités (O:N) pour ses deux rôles.

L'association *Trt-Gen-Msg*

Elle représente le fait qu'un Traitement, à l'instar d'un processus, peut générer des messages. Or, on avait déjà l'association *Trt-Msg-Out*. Ces deux associations possédant les mêmes entités sources et cibles, et qui représentent les mêmes relations entre ces entités, sont redondantes compte tenu qu'un message ne peut être produit en sortie que lors de la terminaison de l'exécution d'un *Traitement*. On ne va donc en garder qu'une seule, soit *Trt-Gen-Msg*.

L'association *Msg-Dec-Trt*

Tout comme l'association *Trt-Gen-Msg*, l'association *Msg-Dec-Trt* correspond à l'association *Msg-Dec-Pro* (un *Message* déclenche un *Processus*).

Un processus devait avoir reçu tous les messages qui lui sont destinés avant de se déclencher. On pourrait dès lors considérer que les associations *Trt-Msg-In* (un *Traitement* reçoit un *Message*) et *Msg-Dec-Trt* sont devenues redondantes à l'instant où l'on unifie les entités *Traitement* et *Processus*.

Or, nous avons vu dans la méthode SADT qu'un *Traitement* (une *Activité* dans SADT) peut recevoir des données en entrée, et ne s'exécuter que lorsqu'il a reçu les données de contrôle nécessaires.

On conserve donc deux associations simplement pour préserver le caractère informationnel et le caractère déclenchant ("contrôle" dans SADT) du message.

Nous allons, en fin de compte, conserver qu'une seule association, soit *Trt-Msg-In* . On lui attribuera le caractère "contrôle" ou "informationnel" via l'attribut booléen *Contrôle*.

L'association Trt-Par-Réa

Cette association est destinée à modéliser le fait qu'un *Traitement* peut participer à une Réalisation. Elle remplace l'association *Pro-Par-Trt*.

L'association Réa-Dec-Trt

Cette association représente le fait qu'une *Réalisation* peut déclencher l'exécution d'un *Traitement*. Elle remplace l'association *Réa-Dec-Pro.*

11.2. Apport de MERISE

Nous allons maintenant tenter d'intégrer dans notre méta-modèle les concepts existants dans MERISE.

11.2.1. Correspondance entre entités

Le tableau ci-dessous donne les correspondances immédiates entre entités du méta-modèle de MERISE et entités de notre méta-modèle.

Il est évident que les entités du méta-modèle de MERISE, qui ne possèdent pas tous les attributs des entités auxquelles on les fait correspondre, prendront pour ces attributs des valeurs "neutres".

Les correspondances marquées d'une astérisque seront commentées par après.

MERISE	Méta-modèle global
Individu-type	Entité-type
Relation-type	Association-type
Propriété-type	Attribut
Identifiant	Identifiant
Message-type	Message-type
Opération-type	Traitement
Synchronisation-type	Réalisation *

Tableau 11.1. Correspondance des entités de MERISE.

Correspondance Synchronisation-type — Réalisation

Une *Synchronisation-type* correspond, évidemment, à une *Réalisation* dont l'attribut *type* à la valeur "Synchronisation".

11.2.2. Correspondance entre associations

De la même manière que pour les entités, nous allons établir une correspondance entre les associations.

MERISE	Méta-modèle global
Ind-Prop	Ent-Att
Ind-Id	Id-Ent
Id-Prop	Id-Att
Ind-Rel	Ass-Rol, Rol-Ent et entité Rôle*
Rel-Prop	Ass-Att
Dépendance Fonctionnelle	Entité Contrainte d'intégrité et associations CI-___*
Type	Nouvelle association: Type*
Prop-Msg	Msg-Att
Compose	Trt-Compose
Déclenche	Trt-Dec-Trt
Op-Eme-Msg	Trt-Gen-Msg
Msg-Dec-Op	Msg-Dec-Trt
Msg-Par-Syn	Msg-Par-Réa
Syn-Eme-Msg	Réa-Gen-Msg
Op-Par-Syn	Trt-Par-Réa
Syn-Dec-Op	Réa-Dec-Trt
Acc-Ind, Acc-Rel, Acc-Prop, Acc-Id	Consultation & Mise-à-jour*

Tableau 11.2. Correspondance des associations de MERISE.

Représentation de l'association Ind-Rel

L'association *Ind-Rel* représente la participation d'une entité-type à une association-type. Cette association *Ind-Rel* possède comme attribut le *rôle* joué et la *cardinalité* de ce rôle.

Nous allons la représenter sous la forme de l'entité *Rôle*, dont l'attribut *connectivité* prendra la valeur de l'attribut *cardinalité* de l'ancienne association. On va de plus employer les associations *Rol-Ent* et *Ass-Rol* pour relier l'entité-type et l'association-type correspondant respectivement à l'individu-type et à la relation-type anciennement associés par Ind-Rel.

Représentation des dépendances fonctionnelles

Dans notre modèle, toute les contraintes d'intégrité sont modélisées grâce à l'utilisation de l'entité *Contrainte d'intégrité* et des associations *CI-Ent*, *CI-Ass*, *CI-Rol*, *CI-Att* et *CI-ID*.

L'entité *Contrainte d'intégrité* représentant la dépendance fonctionnelle possèdera "Dépendance fonctionnelle" comme valeur de son attribut *type* .

Elle sera associée aux *Entités-types*, correspondant aux *Individus-types* respectivement sources et cibles de la dépendance, par des associations CI-Ent dont l'attribut *rôle-joué* possède la valeur "source" et respectivement "cible".

Nouvelle association: *type*

Dans IDA, les Entités-types jouant le rôle de sur-type d'autres entités étaient reliées avec leurs entités-types "filles" par une association-type classique, non définie à priori.

Nous allons, afin de représenter cette notion de manière formelle, comme cela est réalisé dans MERISE, créer une nouvelle association.

L'association *Type* porte sur deux *Entités-types* dont une joue le rôle de "mère", et l'autre celui de "fille".

- Une *Entité-type* peut être "mère" de deux ou plus autres *Entités-types*.
- Une *Entité-type* ne peut être "fille" de zéro, une ou plusieurs (spécialisation) autres *Entités-types*.

Représentation des associations Acc-Ind, Acc-Prop, Acc-Rel & Acc-Id

Ces quatres associations représentaient les accès effectués par les opérations sur les données. Dans notre méta-modèle, elles correspondent aux associations de mise-à-jour et de consultation.

Les associations d'accès dont l'attribut *type* a la valeur "consultation" seront représentée par l'association de consultation correspondant à la cible de la consultation: *Acc-Ind* correspond à *Maj-Ent*, *Acc-Prop* à *Maj-Att*, *Acc-Id* à *Maj-Id*, et *Acc-Rel* à *Maj-Ass* ou *Maj-Rôle* selon le cas.

Les autres associations d'accès seront représentées par les associations de consultation: *Acc-Ind* correspond à *Cslt-Ent*, *Acc-Prop* à *Cslt-Att*, *Acc-Id* à *Cslt-Id*, et *Acc-Rel* à *Cslt-Ass* ou *Cslt-Rôle* selon le cas.

11.3. Apport de REMORA

Pour REMORA, nous allons procéder en considérant les concepts apportés par chaque modèle.

11.3.1. Apport du Modèle descriptif

Les objets

Les objets sont représentés dans un formalisme entité/association. Les correspondances sont dès lors immédiates.

REMORA	Méta-modèle global
t-Objet	Entité-type.
t-Propriété	Attribut
t-Association	Association-type
Identifiant	Identifiant
Prop-Obj	Ent-Att
Prop-Ass	Ass-Att
Ass-Obj	Entité Rôle et associations Ass-Rol et Rol-Ent.
Id-Obj & Id-Prop	Id-Ent & Id-Att
Compose-Obj	Type
Entité CI et associations CI-__	Entité Contrainte d'intégrité et associations CI-__

Tableau 11.3. Correspondance des apports de REMORA.

Les opérations

Une opération correspond intuitivement à un Traitement. Cependant, REMORA permet l'évolution des règles d'action associées aux opérations.

La solution que nous avons choisie est de représenter les *t-Opérations* par les *Traitements* et de leur associer des *Attributs*. Nous créons ainsi, dans notre méta-modèle global, une nouvelle association que nous baptisons *Trt-Att*.

L'association Trt-Att

Elle représente le fait qu'un *Traitement* peut posséder des attributs non permanents.

L'association *Trt-Att* correspond à l'association *Op-Prop* existant dans REMORA.

- Un Traitement peut posséder zéro, un ou plusieurs Attributs.
- Un Attribut peut appartenir à zéro, un ou plusieurs Traitements.

Représentation de l'association modifie

Cette association sera représentée par l'utilisation d'une des associations de mise-à-jour en fonction de "l'objet" modifié. Dans la plupart des cas, il s'agira de l'association *Maj-Att*.

La représentation de l'association Compose-Op est immédiate.

Les Evénements

La notion d'événement est indirectement présente dans notre modèle. En effet, dans IDA et MERISE, un événement est la survenance d'un message, la terminaison d'un traitement, ou la réalisation d'une synchronisation.

Nous n'en avions, dès lors, pas fait une entité à part entière, puisqu'un événement n'avait pas d'existence autonome.

Dans REMORA, il est possible d'associer à un événement des propriétés. La représentation que nous avions adoptée doit donc être modifiée pour le permettre.

L'entité Evénement-Type

On crée l'entité *Evénement-Type*. Un *Evénement-Type* est un ensemble d'événements qui correspondent à la survenance de faits semblables qui engendrent une réaction similaire de la part du système. Un *Evénement-Type* possède un nom qui l'identifie, et une description.

L'association Ev-Att

Cette association permet de donner à un événement-type des propriétés non permanentes. Un *Evénement-Type* pourra posséder zéro, un ou plusieurs *Attributs*.

Association Ev-Msg

Cette association représente le fait que la génération d'un message-type engendre la survenance de l'événement associé. Tout *Evénement-Type* est associé à un et un seul *Message-type*.

Association Ev-Dec-Trt

La survenance d'un événement peut déclencher l'exécution d'un *Traitement*. La connectivité des rôles est (0:N) pour chacun.

Association Trt-Term-Ev

Un *Traitement* en terminaison peut engendrer la survenance d'un événement. Dès lors, on associe un *Traitement* aux *Evénements-Types* qu'il engendre (connectivité des deux rôles: (0:N)).

Association Ev-Par-Réa

Un événement peut participer à la réalisation d'une synchronisation, d'une duplication ou d'une sélection.

- Un *Evénement-Type* peut participer à zéro, une ou plusieurs *Réalisations.*
- Une *Réalisation* est associée à au moins un *Evénement-Type.*

Association Réa-Gen-Ev

La réalisation d'une *Réalisation* engendre la survenance d'un événement.

- Une *Réalisation* génère au moins un *Evénement-Type.*
- Un *Evénement-Type* peut être généré par zéro, une ou plusieurs *Réalisations.*

11.3.2. Apport du modèle conceptuel

Le modèle conceptuel de REMORA apporte les notions de relations (au sens relationnel de CODD). Nous allons donc ajouter à notre méta-modèle global les entités et associations nécessaires pour représenter ces concepts.

Les c-Objets

L'entité R-Entité

Elle correspond en fait à la notion de relation (CODD). Mais le terme "relation" étant ambigu, nous lui avons préféré celui de R-Entité. La lettre "R" rappelant le caractère relationnel de cette entité.

Une *R-Entité* correspond à un c-Objet, et possède:

- un nom
- un type: RP (relation permanente) ou RV (relation variable).
- une description

L'association REnt-Att

Cette association associe des attributs non décomposables aux *R-Entités*, en vue de représenter la relation. Toute *R-Entités* possède au moins un *Attribut*.

Cette association correspond à l'association cObj-Prop.

L'association REnt-Id

Elle permet de représenter qu'une propriété est l'identifiant d'une *R-Entité*. Toute *R-Entité* possède un identifiant. De plus, elle permet de représenter le fait qu'une association sera représentée par une *R-Entité* par une relation (au sens relationnel) possédant comme attribut les identifiants des entités y participant.

Cette association correspond à l'association cObj-ID.

L'association REnt-Ent

Une *Entité-type* correspond à une ou plusieurs *R-Entités*. Cette association permet de le représenter.

Cette association correspond à l'association tObj-cObj.

L'association Ass-REnt

Une *Association-type* sera "représentée" par une *R-Entité*, dont les attributs seront les identifiants des *Entités-types* participant à cette association-type.

Cette association correspond à l'association TAss-cObj.

Représentation des c-Opérations

Une *c-Opération* correspond à un type d'action élémentaire, exécutée en réponse à un *c-Evénement (R-Evénement)*, et modifiant un *c-Objet (R-Entité)*. On lui attribue un nom et un type (Cf. présentation de REMORA).

Ces c-Opérations peuvent sans problème être représentées par des Traitements, à condition qu'il s'agisse de Traitements non décomposables, et que ceux-ci puissent modifier les *R-Entités*.

L'association Trt-Maj-REnt

Une *c-Opération*, représentée par un *Traitement*, peut être associée à une *R-Entité* sur laquelle portera son action.

- Un *Traitement* peut porter sur une seule *R-Entité*.
- Une *R-Entité* peut être modifiée par zéro, un ou plusieurs *Traitements*.

L'association Trt-cOp-tOp

Une *c-Opération*, représentée par un *Traitement*, est associée à une t-Opération, elle aussi représentée par un *Traitement.*

Pour représenter cette association, un *Traitement* sera relié à un autre Traitement par l'association Trt-cOp-tOp.

Représentation des c-événements

L'entité c-Evénement

Un *c-Evénement* correspond à un type de changement d'état élémentaire du *c-Objet* associé. On attribue, en plus d'un nom, un type à cette entité .

Un *c-Evénement* peut être représenté par un *Evénement-Type*. On lui attribuera en plus un *type* (REV, RPRED, RCONDFAC, REVARR, REVDECL, ou Classique s'il ne correspond pas à un c-Evénement).

L'association tEv-cEv

On met en relation chaque *t-Evénement* et le ou les *c-Evénements* qui permettent de le représenter sous forme normalisée.

Dès lors, on crée dans notre méta-modèle une association *tEv-cEv* qui relie les *Evénements-Types* entre eux, et correspond à la même association entre *t-Evénéments* et *c-Evénéments.*

L'association Ev-Att

Cette association permet de mettre en relation les c-Evénements et les Propriétés qui les composent. Elle correspond à l'association cEv-Prop définie dans le méta-modèle de REMORA.

L'association Constate

On représente ainsi l'association d'un c-Evénement, représenté par un *Evénement-Type*, avec le seul c-Objet (*R-Entité*) dont il constate un changement d'état.

L'association Déclenche-cOp

Tout c-Evénement peut déclencher une ou plusieurs c-Opérations. Cette association possède en attribut les éventuels facteur et condition de déclenchement.

Cette association sera représentée par le déclenchement direct d'un Traitement par un Evénement-Type, soit via l'utilisation d'une *Réalisation* permettant de prendre en considération la condition attribuée à cette association.

11.4. Apport de OOA

OOA	Méta-modèle Global
Entité *Objet*	Entité *Entité-type*
Entité *Attribut*	Entité *Attribut*
Entité *Relation*	Entité *Association-type, Rôle* et association *Ass-Rol* *
Association *Att-Obj*	Association *Ent-Att*
Association *Att-Rel*	Association *Ass-Att*
Association *Rel-Obj*	Association *Rol-Ent*
Entité *Identifiant*	Entité *Identifiant*
Association *Id-Obj*	Association *Id-Ent*
Association *Id-Att*	Association *Id-Att*
Entités *Table de corrélation* & *Objet Associatif*	Entité *R-Entité*
Associations *Rel-Tc* et *Rel-OA*	Association *Ass-REnt*
Association *Tc-Id* et *OA-Id*	Association *REnt-Id*
Association *Att-OA*	Association *REnt-Att*
Association *Sur-type*	Association *Type*
Entité *Attribut-Status*	Entité *Attribut* *
Association *Obj-Sta*	Association *Ent-Att*
Entité *Transition*	Entité *Transition* *
Association *Trans-Sta*	Association *Trans-Att-Sta* *
Entité *Evénement*	Entité *Evénement*
Association Ev-Obj	Association *Ev-Des-Ent* *
Association Ev-Trans	Association *Ev-Pro-Trans* *
Entité *Opération*	Entité *Traitement*
Association *Op-Trans*	Association *Trt-Trans* *
Association *Op-Ev*	Association *Trt-Term-Ev*

Entité *Flux*	Entité *Message-type*
Association *Flux-Out-Op*	Association *Trt-Gen-Msg*
Association *Flux-In-Op*	Association *Trt-Msg-In*
Association Att-Flux	Association *Msg-Att*
Associations *Accède*	Associations *consultation* et *mise-à-jour*.
Associations *Compose*	Associations Trt-Compose
Association *Déclenche*	Associations Trt-Dec-Trt

Tableau 11.4. Correspondance des apports de OOA.

Représentation de l'entité *Relation*

Elle va être représentée par une association-type. Cependant une *Relation* possède comme attribut sa multiplicité. Nous allons donc devoir convertir cette multiplicité afin d'obtenir les connectivités associées à chaque rôle.

Le tableau suivant donne, pour chaque valeur de multiplicité, les connectivités correspondantes.

Multiplicité	Connectivité du premier rôle	Connectivité du second rôle
(1:1)	(1,1)	(1,1)
(1:1c)	(0,1)	(1,1)
(1c:1c)	(0,1)	(0,1)
(1:M)	(1,N)	(1,1)
(1c:M)	(1,N)	(0,1)
(1:Mc)	(0,N)	(1,1)
(1c:Mc)	(0,N)	(0,1)
(M:M)	(1,N)	(1,N)
(M:Mc)	(0,N)	(1,N)
(Mc:Mc)	(0,N)	(0,N)

Tableau 11.5. Correspondance entre Connectivités et Multiplicités.

On étendra facilement ces correspondances à des relations ternaires et de degré supérieur.

L'entité Attribut-Status

Elle possédait une existence autonome dans le méta-modèle de OOA, mais cela ne se justifie que pour des raisons de clarté. Elle sera représentée dans le modèle global par une entité *Attribut*.

L'entité Transition

Une transition correspond à la modification de l'attribut status associé à un objet. Une transition est définie par un nom, une valeur de départ et une valeur d'arrivée.

L'association Trans-Att-Sta

Cette association représente le fait que toute *Transition* porte sur un et un seul *Attribut* (status)

L'association Ev-Des-Ent

Un événement peut être destiné à zéro ou une *Entité-type.*

L'association Ev-Pro-Trans

Un événement peut provoquer le déclenchement d'une *Transaction.* Toute *Transaction* doit être déclenchée par un *Evénément-Type.*

L'association Trt-Trans

On représente par cette association le fait que un ou plusieurs *Traitments* peuvent être associé à une *Transition* . Auquel cas, ces *Traitements* seront exécutés lors du déclenchement de la *Transition.*

11.5. Apport de SADT

La correspondance entre les éléments du méta-modèle de SADT et ceux de notre modèle est quasi triviale.

SADT	Méta-modèle Global
Entité *Activité*	Entité *Traitement*
Association *Compose-Act*	Association *Trt-Compose*
Entité *Donnée*	Entité *Message-type* *
Association *Compose-Don*	Association *Msg-Att*
Entité *Flux*	Entité *Message-type*
Association *Don-Flux*	Association *Msg-Att*
Association *Flux-In-Act*	Associations *Trt-Msg-In* *
Association *Flux-Out-Act*	Associations *Trt-Gen-Msg.*
Association *Influence*	Association Trt-Ver-CI*

Entité *Ressource*	Entité *Ressource*
Association *Mécanisme*	Association *Réquisition*
Association Support	Association *Support* *

Tableau 11.6. Correspondance des apports de SADT.

Les entités Données

On a deux possibilités pour les représenter:

- Soit on les représente par les entités *Entité-type*, *Attribut-type*, *Association-type*, *Rôle* , et *Identifiant* qui correspondent le mieux à la *Donnée* représentée.
- Soit on considère que ces données sont véhiculées entre les activités par des messages, et l'on choisit pour les représenter l'entité *Message-type*.

Nous avons opté pour cette seconde solution, car elle représente mieux le fait que les *Données* circulent entre les activités.

L'Association Flux-In-Act

Trt-Msg-In avec comme valeur de l'attribut *contrôle* "Vrai" s'il s'agit d'un contrôle ou "faux" s'il s'agit d'un entrée.

L'association Influence

Cette association représente le fait qu'une activité influence la création et les manipulations de la donnée influencée. Cela revient, en fait, à dédier une activité pour le contrôle de contrainte d'intégrité.

Nous allons la représenter par une nouvelle association, nommée *Trt-Ver-CI*, entre les Traitements et les Contraintes d'intégrité.

- Un *Traitement* peut vérifier zéro, une *Contrainte d'intégrité.*
- Une *Contrainte d'intégrité* peut être vérifiée par zéro, un ou plusieurs *Traitements.*

L'association Support

Cette association représente le fait qu'une donnée se trouve sur un support particulier. Nous l'introduisons dans notre modèle, en temps qu'association entre les entités *Message-type* et *Ressource.*

- Un *Message-type* peut être "supporté" par zéro ou une *Ressource.*
- Une *Ressource* peut être le support zéro, un ou plusieurs *Messages-types.*

11.6. Représentation graphique

11.6.1. Représentation des données

Notre modèle Entité/association de base n'a que peu été modifié. On y a ajouté l'association *Type* entre entités-types.

De plus, afin d'intégrer les notions relationnelles, on a également ajouté l'entité *R-Entité* et les associations qui s'y rapportent. Nous aurions pu considérer les R-entités comme des Entités-types, mais nous aurions dans ce cas perdu le caractère relationnel. Nous aurions pu également les considérer comme un sous-ensemble d'entités-types.

Afin d'alléger le graphique, nous avons scindé le modèle en deux parties: une reprenant les concepts du modèle entité/association, et l'autre les notions relationnelles. Nous obtenons donc les représentations graphiques suivantes:

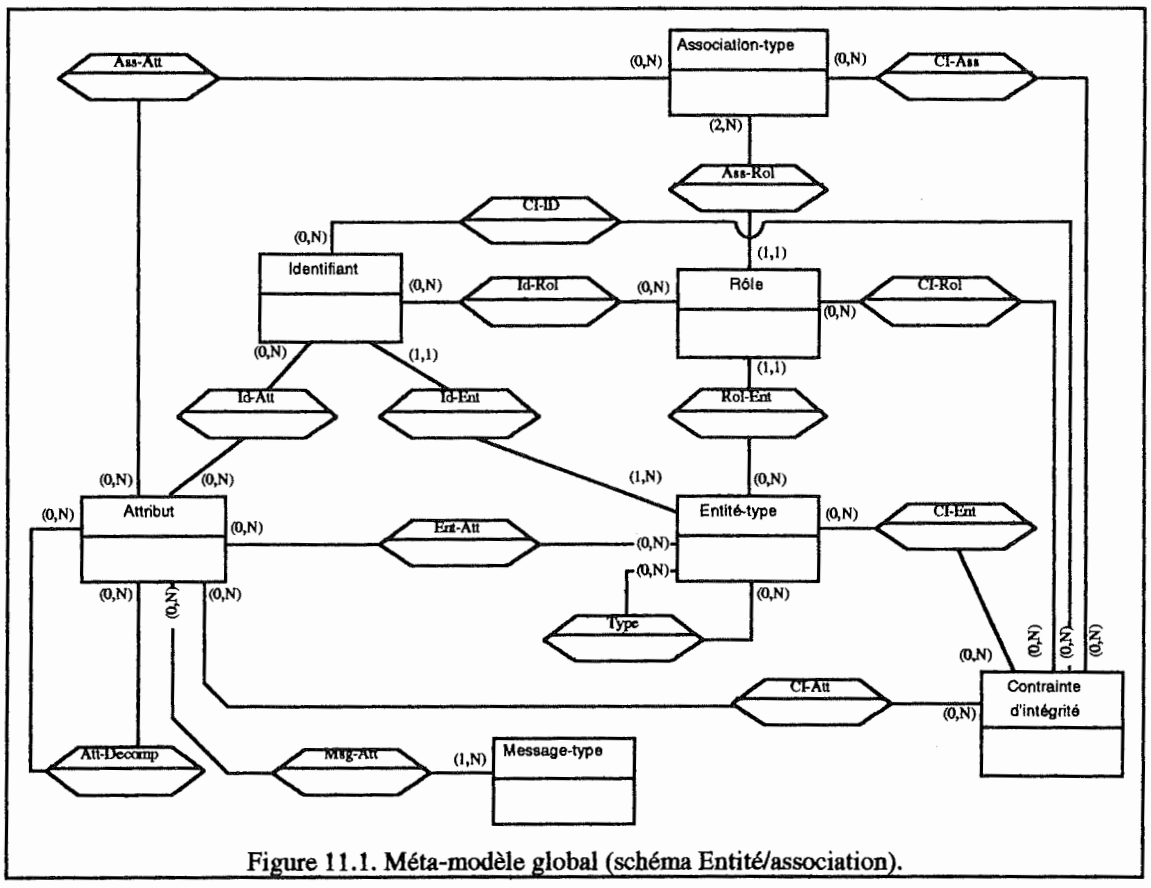


Figure 11.1. Méta-modèle global (schéma Entité/association).

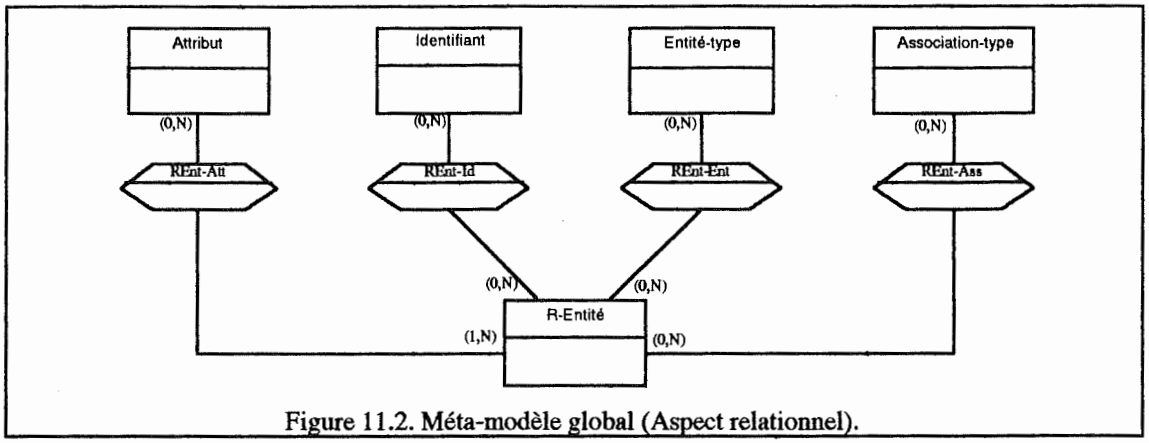


Figure 11.2. Méta-modèle global (Aspect relationnel).

11.6.2. Représentation des traitements

Nous avons rencontré deux classes de notions concernant les traitements:

- notions statiques: décomposition hiérarchique, accès aux données, entrée/sorties de messages, utilisation de ressources.
- notions dynamiques: déclenchement, événements,...

La représentation graphique de ces concepts est présentée au figure 11.3 et 11.4 ci-dessous.

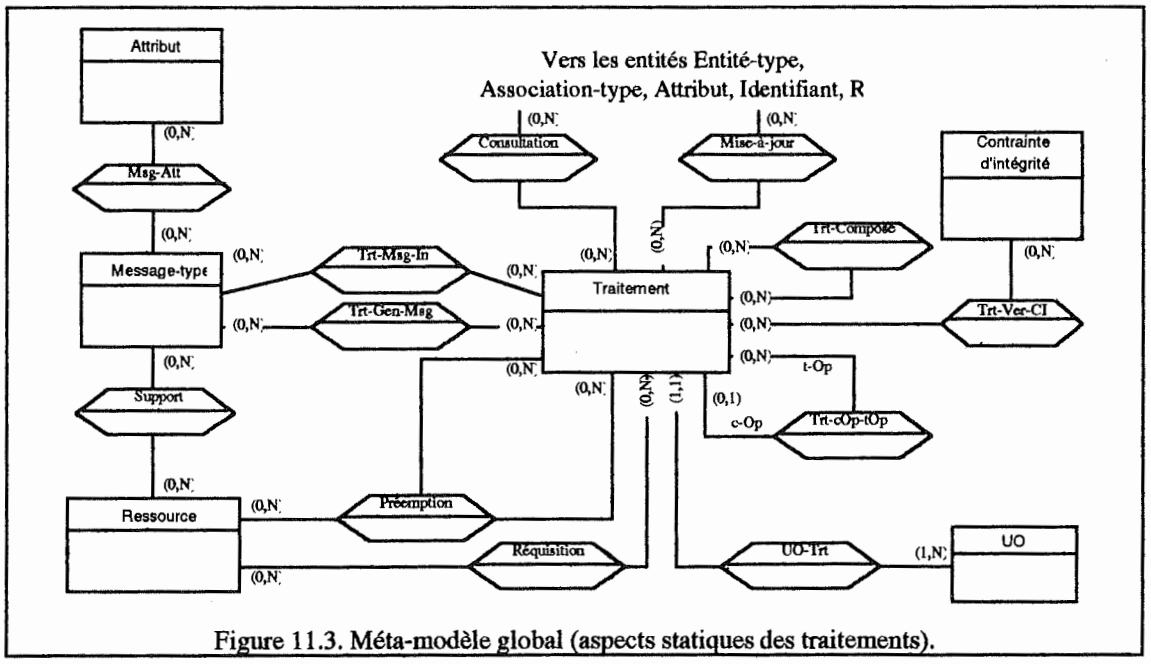


Figure 11.3. Méta-modèle global (aspects statiques des traitements).

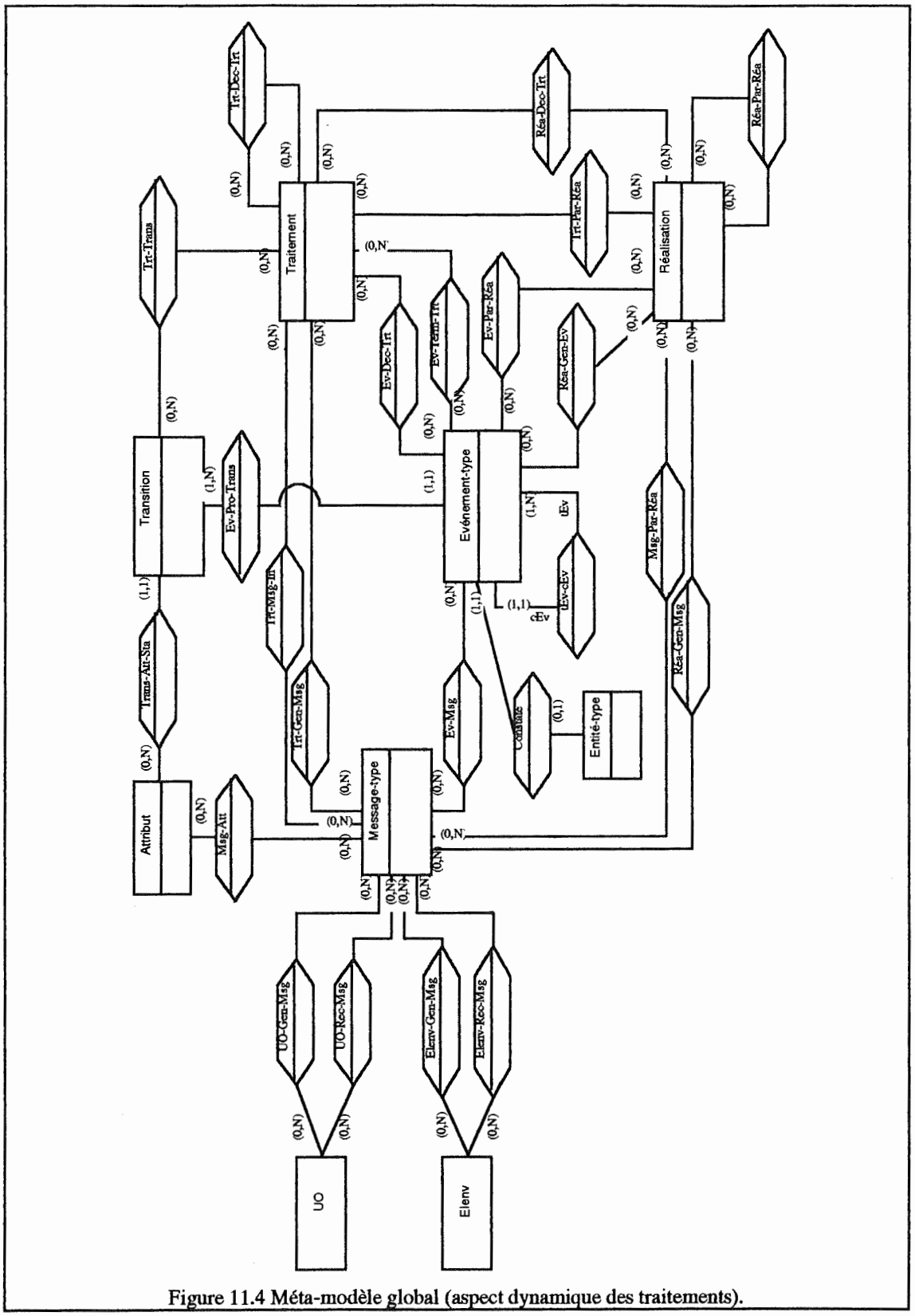


Figure 11.4 Méta-modèle global (aspect dynamique des traitements).

Partie III

Conception d'outils

12. Conception d'outils

L'intérêt croissant des organisations pour les méthodes de conception de système d'information se manifeste par divers signes:

- investissements en matière d'exploitation de ces méthodes,
- développement d'outils les mettant en œuvre
- recherche et embauche d'informaticiens maîtrisant ces méthodes.

Ces divers éléments ont entraîné le développement des ateliers de génie logiciel, et de divers programmes d'interface, dont LCI.

Le méta-modèle élaboré, dans ce travail est l'extension du travail réalisé lors de mon stage de fin d'étude.

Durant ce stage, il m'était demandé de fournir un méta-modèle reprenant les concepts destinés à la représentation du comportement dynamique des systèmes d'information. Ces concepts étaient ceux présent dans les méthodes mais aussi, voire même principalement, ceux contenus dans les outils CASE.

Le but visé était d'utiliser le méta-modèle élaboré pour spécifier le dictionnaire d'interface d'une future version 2.0 du programme LCI (Cf. infra).

12.1. Les "Ateliers de Génie Logiciel"

Un atelier de génie logiciel (AGL) est un ensemble d'éléments de natures diverses destinés à permettre aux développeurs "de formaliser, de décrire, et de maîtriser l'ensemble des activités qui concernent le développement du logiciel"[UNI. GM].

"Le génie logiciel regroupe toutes les activités dont l'objectif est que le développement soit une activité qui relève plus de l'ingénierie que de l'art."[UNI. GM]

Les AGL sont constitués de méthodes et d'outils.

12.1.1. Les méthodes

Il peut s'agir de méthodes de conception, telles que MERISE et SADT, destinées au développement à proprement parler du logiciel.

On dispose également de méthodes destinées aux activités "annexes" du développement. Il peut s'agir de méthodes:

- d'estimation et de prévision des charges: estimation du volume de code, prévision des charges de travail, des délais

- de conduite de projet et de gestion des ressources: planification des activités, identification et gestion des ressources critiques.
- de documentation
- de gestion de configurations

12.1.2. Les outils

Repris sous le nom générique de CASE, pour "Computer Aided Software Engineering", on en distingue deux catégories: les outils de réalisation et les outils.

Les outils de réalisation

Les outils de réalisation forment une couche supplémentaire au dessus des systèmes d'exploitation, et permettent aux développeurs un accès plus aisé aux ressources de la machine.

Ces outils sont regroupés sous le terme générique de "LOWER CASE". On retrouve notamment dans cette catégorie:

- Les éditeurs, compilateurs, et débuggueurs associés à des langages de programmation.
- Les systèmes et langages de gestion de bases de données (DBMS, SQL, ...).
- Les gestionnaires d'écrans.

Les outils de conception

Ces outils sont des logiciels qui permettent aux développeurs d'automatiser certaines activités inhérentes à la méthode choisie.

Plusieurs rôles sont joués par ces logiciels:

- Assister les concepteurs lors de l'élaboration des modèles et des schémas.
- Contrôler (en partie) automatiquement les modèles élaborés.
- Gérer de manière homogène les divers descriptions réalisées.
- Centraliser tout ce qui a été dit au cours du cycle de vie du logiciel.
- Faciliter les modifications des descriptions.
- Faciliter le transfert d'informations entre différents modèles.

Ces outils sont regroupés sous le terme générique de "UPPER CASE". Ils sont soit dédiés à une méthode précise (IDA, MEGA[9]), soit ouverts (associés à aucune

[9] IDA™ de METSI associé à la méthode de même nom; MEGA de GAMMA Int'l, associé à MERISE.

méthode particulière), ou bien encore paramétrables, c'est-à-dire qu'ils peuvent être configurés de manière à s'adapter à la méthode choisie (EXCELERATOR[10]).

12.1.3. Avantages des AGL

L'utilisation d'ateliers de génie logiciel présente, outre l'utilité de l'emploi d'une méthode, divers avantages.

• L'utilisation d'un CASE permet une formalisation assistée des spécifications du système, grâce aux méthodes théoriques associées à l'AGL , ainsi qu'un contrôle de cohérence en partie automatisé de ces spécifications.

• Cette formalisation est d'autant plus aisée que les AGL offrent maintenant des outils de conception, tant graphiques que textuels, performants et faciles à utiliser dans un environnement "micro" convivial.

Les résultats de cette formalisation peuvent aisément être transmis et compris par les utilisateurs.

• Libérant donc les concepteurs des tâches "mécaniques", les AGL leur permettent de se consacrer entièrement aux spécifications qui sont la base de tout développement informatique.

Tout ceci confère un caractère plus communicable aux documents de base, et notamment au cahier des charges, qui devient par là-même, clair, complet et formel.

• Les AGL permettent la maintenance plus aisée des systèmes développés en assurant une documentation de ceux-ci lors de toutes les étapes de leur cycle de vie.

Cette documentation présente l'avantage d'être constamment mise à jour en fonction des modifications apportées, via l'AGL, au niveau des spécifications. Cela permet une cohérence quasi-automatique entre les différents niveaux de développement (spécifications, conception générale, conception détaillée, codage).

Ce "suivi" de la documentation permet un gain de temps appréciable lors de l'élaboration d'une documentation finale destinée aux utilisateurs et au personnel de maintenance du programme.

---

[10] EXCELERATOR est un produit de INDEX TECHNOLOGY.

12.1.4. Conséquences économiques

De par ces faits, l'utilisation D'AGL génère au niveau des produits développés une augmentation de leur qualité et de leur conformité par rapport aux spécifications.

Cette augmentation de qualité et de conformité engendre, en plus d'une réduction de l'effort de développement et de maintenance du système, une augmentation de la rentabilité-même des équipes de conception, de réalisation, et de maintenance.

Les programmes et systèmes deviennent donc moins coûteux pour les entreprises, mais également plus aisés à concevoir.

Il apparaît cependant que l'emploi d'AGL, et des outils associés, reste encore assez restreint. Leur coût d'acquisition élevé en est un frein, face à un gain non chiffrable à priori.

De plus, au coût d'achat d'un AGL viennent s'ajouter les coûts, non négligeables, de formation du personnel, car l'utilisation d'AGL implique les contraintes inhérentes à l'emploi d'une méthode de conception.

Même si tout le monde informatique s'accorde pour en vanter les vertus, ces méthodes de conception de système d'information sont encore peu utilisées.

Cependant, de plus en plus d'entreprises franchissent le "cap de la méthode", et ouvrent ainsi de nouvelles perspectives en matière de génie logiciel.

12.2. LINC Case Interface

12.2.1. Introduction

L'utilisation, néanmoins croissante, d'outils CASE pour la conception des systèmes d'information a amené le développement du produit LCI chez UNISYS.

LCI permet aux développeurs d'applications en langage LINC (Cf. infra) d'utiliser les UPPER-CASE, dans un environnement PC pour la conception, la validation, et la documentation de leurs applications, et d'en récupérer les résultats sur mainframe UNISYS.

Le principal intérêt de cette démarche, outre le fait de l'emploi d'outils CASE, est de récupérer le résultat des étapes de conception, sans avoir à le réintroduire manuellement.

Cette façon de faire évite non seulement le "copiage" manuel du travail déjà réalisé, mais aussi les erreurs et les oublis possibleslors de l'étape de transfert, d'où gain de temps et diminution des risques.

De plus, toute modification apportée dans le modèle généré dans le CASE sera répercutée dans la base LINC, alors qu'auparavant, il était plus difficile de maintenir la cohérence entre la base LINC et la base du CASE, puisque l'on devait reporter les modifications dans les deux bases.

Enfin, aucune restriction n'est imposée quant au CASE à employer: libre choix aux concepteurs donc...

Alors que les CASE-Tools "acceptés" sont d'origines diverses, LCI est développé par UNISYS dans son centre de développement français de Val de Reuil. L'initiative du développement de LCI est française, bien que le développement du produit dépende directement de la maison mère (UNISYS Corp. - USA).

Quant à LINC, il s'agit d'un langage de quatrième génération développé par BURROUGH pour ses mainframes, et transposé aux machines SPERRY depuis la fusion BURROUGH-SPERRY qui a donné lieu à la création d'UNISYS.

12.2.2. Principes de fonctionnement

Démarche globale

Le produit LCI se décompose en plusieurs programmes : Les BRIDGES, CAPTURE et UPLOAD. La démarche se déroule en plusieurs étapes:

• L'utilisateur élabore divers modèles dans le CASE de son choix, sur PC.

• Il sauvegarde son travail dans la base de données propre au CASE, aussi appelée dictionnaire (ou "repository").

• Il lance l'exécution du programme de transfert correspondant à son CASE. Ce programme porte le nom générique de BRIDGE. Le BRIDGE crée un fichier dans un format intermédiaire (I.L.C.I).

• L'utilisateur, toujours sur PC, lance le programme CAPTURE qui procède au transfert du contenu des fichiers interfaces vers une base de données propres à LCI.

• L'utilisateur, lance l'exécution du programme UPLOAD. Ce programme (sur PC) met en relation un PC et un mainframe UNISYS, et crée une session de travail en langage LINC sur le mainframe. Lors de cette session, le programme va transférer le contenu de la base LCI vers la base LINC.

• Enfin, l'utilisateur ouvre une session LINC sur le mainframe, et constate que le contenu de la base du CASE a bien été transféré.

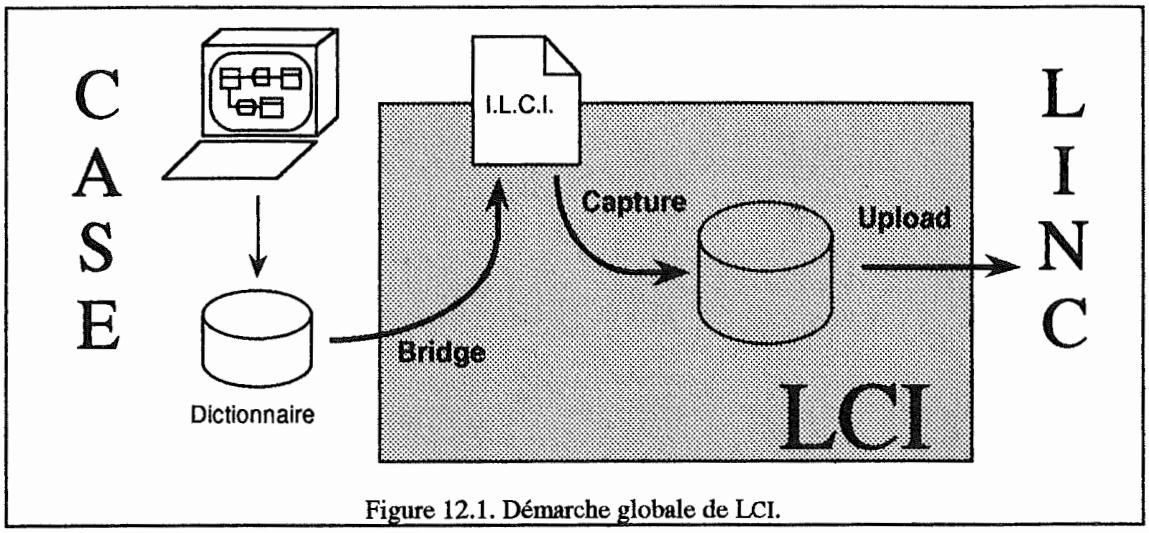


Figure 12.1. Démarche globale de LCI.

Les BRIDGES

A chaque CASE correspond un bridge particulier. Les bridges extraient, du dictionnaire des CASE-Tools associés, divers éléments et les transfèrent dans un fichier d'interface.

Les éléments extraits sont, dans la première version de LCI :

- Les modèles conceptuels de données: modèles reprenant la définition conceptuelle des données indépendamment de la structure de la base de données : entités, attributs, associations, contraintes d'intégrité,...
- Les modèles logiques de données, dépendant de la structure de la base de données: record, champs, clés d'accès.
- Le dessin des écrans: libellés, zones de saisie d'information, zones répétitives.
- Les commentaires associés à ces éléments.

La seconde version de LCI permettra d'extraire, en plus de ces éléments, les spécifications des traitements élaborées dans le CASE. On extraira donc :

- La définition des traitements : description, règle (algorithme), actions sur la base de données.
- Les éléments de la base de données accédés, et la définition de ces accès.
- Les relations statiques entre traitements : composition
- Les relations dynamiques entre traitements: déclenchement, appel ainsi que les paramètres passés et reçus.

La technique d'extraction utilisée reste, en règle générale, la même quel que soit le CASE employé. L'extraction se fait en deux temps.

Tout d'abord, dans le CASE, on exécute un utilitaire d'élaboration de rapport. Cet utilitaire crée un fichier texte contenant les informations désirées. Le format de ce rapport est propre au CASE et contient divers mots clés permettant d'identifier son contenu.

Ensuite, le programme BRIDGE transforme ce rapport en un fichier au format I.L.C.I (Interface LCI). Dans ce format, les premiers caractères de chaque ligne identifie le type du contenu de la ligne.

```
...
[INFORMATION:        Mot clé signifiant qu'il s'agit d'une information
NOM-CLI                    ( attribut dans MEGA)
Nom de famille       Commentaire associé à l'information
[CLASSE: A           Type de l'attribut : A = Alphanumérique.
[LONGUEUR: 20        Longueur de l'attribut : 20 caractères.
...
[RECORD:             Mot clé signifiant qu'il s'agit d'un
CLIENT                enregistrement de la base de données.
[INFO:               Le record Client possède l'attribut NOM-CLI.
NOM-CLI
[CLE:                NUM-CLI est une clé d'index pour Client et
NUM-CLI              un attribut de Client.
...
```

Figure 12.2. Exemple de contenu d'un fichier extrait grâce aux dossiers internes du case Mega.

Ce type de fichier est donné en entrée au bridge.

Le BRIDGE modifiera certains éléments en vue de leur introduction dans LINC. Par exemple, une information décomposable ne se retrouvera pas dans le fichier d'interface; l'adresse se décomposant en un nom de rue, un numéro et une ville, ne sera pas transmise telle quelle : seul ses composantes seront transférées puisque, pour LINC, seules les informations de bas niveau sont pertinentes. On n'enregistre dans l'interface que les informations non décomposables. Il en est de même pour les attributs des records.

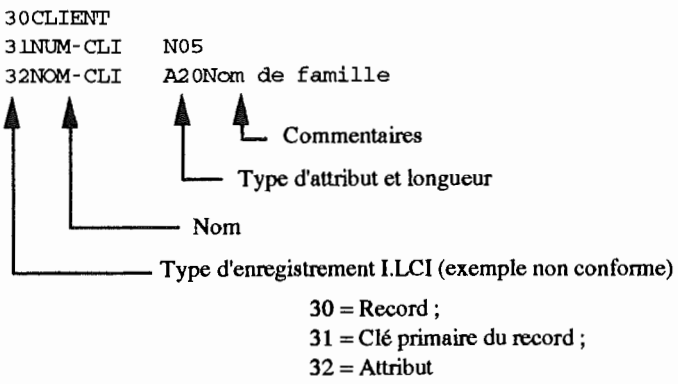


Figure 12.3. Exemple de contenu d'un fichier au format ILCI.

Remarques concernant les bridges:

Les bridges n'étant pas développés par UNISYS, et afin d'en facilité la création et la maintenance, certaines décisions ont été prises en vue de les simplifier :

- Les bridges n'effectuent aucun contrôle quant à la validité des informations qu'ils traitent.
- Les bridges n'accèdent pas directement à la base LCI. Cet accès est laissé au programme CAPTURE. Cela permet notamment de conserver une totale liberté en ce qui concerne le format de la base propre à LCI.

Il est évident que certains CASE-Tools, offrant des utilitaires d'extraction suffisamment puissant, pourraient ne pas recourir à un programme BRIDGE externe au CASE.

CAPTURE

Transfert et vérifications

Ce programme sert d'abord au transfert du contenu des fichiers interfaces produits par les différents BRIDGES vers la base de donnée propre à LCI. Cette étape n'est pas intégrée aux BRIDGES afin de conserver leur simplicité.

Le programme CAPTURE va effectuer plusieurs vérifications sur les éléments importés afin de voir s'ils n'entrent pas en conflit avec les possibilités de LINC. En effet, divers contrôles de cohérence entre ces éléments ont été réalisés dans le CASE. Il s'agit maintenant de vérifier leur conformité par rapport aux contraintes qu'impose le langage LINC.

D'autres rôles sont confiés au programme CAPTURE:

Permettre à l'utilisateur de sélectionner les éléments à transférer

Dans le programme CAPTURE, l'utilisateur va pouvoir sélectionner les éléments, extraits du CASE, qui devront réellement être transférés vers le mainframe.

Cela permet de ne retransférer vers LINC que les éléments ayant subis, dans le CASE, des modifications, et ainsi de gagner du temps: le transfert de la totalité des informations extraites du CASE, entre PC et mainframe, est assez lent pour de gros volume.

Cette sélection des éléments à retransférer aurait dû idéalement être réalisée lors de l'extraction par le BRIDGE. Malheureusement, les BRIDGES ne sont pas tous capables de telles actions, et sont, de plus, développés par des tiers (en général les

développeurs des CASE-Tools), peut soucieux de performances. C'est pour cette raison que le rôle de sélection est confié au programme CAPTURE.

Création de PROFILES et d'ISPECS

De plus, CAPTURE permet de définir certaines relations propres au langage LINC entre les éléments de la base. Il s'agit notamment des PROFILES et des ISPECS.

Les PROFILES

En simplifiant on peut dire qu'un PROFILE correspond à un accès privilégié vers un enregistrement de la base de données (accès semblable à un index). Cet accès doit exister pour accéder de manière non séquentielle à la base de données.

Chaque PROFILE, une fois créé, est entièrement géré par le langage LINC, et est employé grâce à de multiples instructions de ce langage.

Les ISPECS

Les ISPECS correspondent à une entité LINC qui associe un enregistrement de la base de données à un écran. Un ISPEC assure la gestion (création, modification, suppression) de l'enregistrement via cet écran et le code associé par l'utilisateur à l'ISPEC.

Par exemple, supposons que notre base de données contienne l'enregistrement *client* et un écran permettant de saisir des valeurs pour les champs de cet enregistrement. Par le simple fait d'associer l'écran et l'enregistrement via un ISPEC, LINC va générer automatiquement tout le code nécessaire à la gestion de l'enregistrement *client* par l'écran choisi.

L'association (BIND) d'écran et de record pour créer des ISPECS est réalisée dans CAPTURE, au travers d'un dialogue entre le programme et l'utilisateur. L'association d'un traitement à un ISPEC est envisagée pour la seconde version de LCI.

Une fois que l'on a sélectionné les éléments qui seront à transférer vers LINC, et que les ISPECS sont définis, on va pouvoir exécuter le programme UPLOAD.

UPLOAD

Le programme UPLOAD est chargé du transfert des données enregistrées dans la base LCI vers la base LINC. Pour ce faire, UPLOAD va ouvrir une session interactive avec LINC (sur site central UNISYS).

L'éditeur du langage LINC envoie à l'utilisateur (en l'occurrence le programme UPLOAD) différents écrans[11], celui-ci les reconnait, les remplit en utilisant les informations contenues dans la base LCI, et les renvoie à LINC.

L'intérêt de cette démarche, par opposition au transfert d'un fichier au format LINC du PC vers le mainframe, est que le programme UPLOAD ne doit pas vérifier la syntaxe du fichier transféré. En effet, l'éditeur du langage LINC se charge de vérifier les données qui lui sont transférées, comme il le ferait lors de toute session normale.

Outre une simplification importante du programme UPLOAD, cela permet également d'effectuer automatiquement des contrôles de cohérence par rapport à la base existante sur le mainframe.

En fin de session d'UPLOAD, l'utilisateur peut consulter un fichier reprenant les messages reçus de LINC et précisant les éventuelles erreurs survenues. L'utilisateur pourra dès lors corriger sa base sur PC et ne recharger que les éléments erronés.

12.2.3. Intérêt de la démarche LCI

Par opposition à la démarche qui aurait été d'imposer un CASE aux développeurs LINC, la solution choisie par UNISYS paraît plus appropriée.

En effet, le choix d'un outil CASE s'effectue en fonction, non seulement des possibilités de ce CASE, mais également, voire même principalement, en fonction de la méthode de conception associée à ce CASE.

Or, choisir une méthode est bien plus qu'une question de performance d'outil. Il s'agit d'adhérer à la démarche ainsi qu'à la vision du système d'information proposée par les concepteurs de la méthode. Ici interviennent des critères culturels, d'éducation, ...

Pour cette raison, il semble plus judicieux de laisser à chacun le choix de la méthode de spécification et de l'outil CASE, et de fédérer ces spécifications en vue de les transposer vers le langage LINC.

Outre le fait de laisser une grande liberté aux développeurs quant au choix du CASE qu'ils utiliseront, cette solution permet aux sociétés désireuses de s'équiper en moyens informatiques de récupérer l'existant, et ce, quel que soit le CASE employé auparavant. De plus, dans cette même optique, le coût d'équipement est diminué du coût de la formation du personnel à l'emploi d'un nouvel outil CASE.

---

[11] Les terminaux UNISYS fonctionnent en mode page.

Enfin, cela permet aux développeurs utilisant déjà le langage LINC de se tourner vers les outils CASE sans risque d'augmenter leur charge de travail par un recopiage inutile des données du CASE-Tool vers LINC.

Bref, LCI apparaît comme un atout supplémentaire dans la chaîne de développement de logiciel en langage LINC.

12.3. Au-delà de LCI

On remarque que la plupart des CASE-TOOLS offre déjà ce genre d'interface vers d'autres langages, et en particulier vers le langage COBOL.

Indépendamment d'UNISYS et de LINC, l'approche LCI n'apporterait-elle donc rien de nouveau ?

La démarche présentée par UNISYS va bien au-delà de ce qu'offrent les CASE-Tools. En effet, le principe de LCI est de proposer un format d'interface entre les outils CASE existants et un langage. On pourrait imaginer deux développeurs utilisant chacun un CASE différent et fédérant le résultat de leur travaux en une seule application informatique.

Par contre, LCI ne permet pas le passage d'informations d'un CASE vers un autre. Pour ce faire, il faudrait un format standard, non seulement entre les CASE-Tools et un programme, mais également entre les outils CASE eux-mêmes.

Cette idée d'un format intermédiaire et commun aux CASE-Tools et à d'autres applications est reprise, entre autres, dans les annonces de "REPOSITORY" d'IBM.

L'intérêt d'un tel "dépôt" standard est évident. Quel que soit le CASE utilisé, quelle que soit la méthode de spécification employée, il serait possible d'en récupérer le résultat, de le modifier au besoin, et, d'en dériver une application. On pourrait ainsi passer de n'importe quel outil CASE vers n'importe quel langage.

Enfin, mais ce n'est pas encore pour tout de suite, on pourrait, en grande partie, automatiser les phases de codage pour en arriver, un jour, à "compiler des spécifications".

13. Conclusion

L'objectif de notre travail était de réaliser un méta-modèle reprenant les concepts définis dans diverses méthodes de conception de système d'information.

Ce méta-modèle étant une contribution à l'élaboration d'une interface entre outils de conception et outils de réalisation.

Pour ce faire, nous avons donné un aperçu des méthodes MERISE, IDA, REMORA, OOA et SADT, puis nous avons élaboré pour chacune de ces méthodes un méta-modèle.

Nous avons à partir de ces cinq méta-modèles bâti un méta-modèle global permettant de représenter tous les concepts définis dans les méthodes de conception étudiées.

De ce méta-modèle, nous pouvons tirer les conclusions suivantes quant à la représentation des éléments constitutifs des systèmes d'information:

13.1. Représentation des données

L'élaboration de notre méta-modèle global, nous a permis de voir que le modèle entité/association, ou les modèles assimilés (CHEN), était utilisé par la quasi-totalité des méthodes abordées.

Ce formalisme semble être un standard de fait, puisque des méthodes n'incluant pas de modèle de description des données, telles que SADT, sont complétées par ces modèle [LIS. 90].

Le formalisme relationnel de CODD est lui aussi devenu un standard. Il est l'étape "obligée" entre le formalisme entité/association et l'implémentation des données.

Ce formalisme permet une représentation des données sous une forme élémentaire, non décomposable, qui minimise le nombre d'inter-relations entre les éléments. Il permet de relever aisément les éventuelles incohérences, contradictions, ou redondances, et fournit une représentation minimale de l'ensemble des données à traiter. [Cod. 70] [Rol. 88] [Roc. 89]

Les modèles entité/association et relationnel sont donc les deux modèles les plus utilisés pour la modélisation des données.

On remarque que pour la plupart des utilisateurs de méthodes de conception, la conception se limite aux données. Les modèles proposés pour la modélisation des traitements sont peu ou prou utilisés.

13.2. Représentation des traitements

Si les modèles entités/associations et relationnel ont su s'imposer pour la représentation des données, en ce qui concerne les traitements chaque méthode propose une vision particulière du problème.

On a cependant remarqué l'utilisation de deux classes de concepts: les concepts statiques et dynamiques, et pour ceux-ci trois familles de modèles.

13.2.1. Concepts statiques

Les concepts statiques regroupent les éléments définissant les traitements, leur localisation, ainsi que leur relations avec l'extérieur, et ce indépendamment de leur comportement.

On retrouve:

- la définition des traitements,
- la structure hiérarchique ou en réseau des traitements,
- la description de leur contenu, sous une forme variable: texte libre, pseudo-code, expression fonctionnelle,
- la description des messages (informations) reçus en entrée et produits en sortie,
- la description des accès aux données (mise à jour, consultation),
- les contraintes sur les données que les traitements doivent vérifier
- les ressources nécessaires à leur exécution,
- l'unité organisationnelle responsable du traitement

13.2.2. Concepts dynamiques

Les concepts dynamiques permettent de représenter le comportement des traitements et leurs relations dynamiques (enchaînement). Plusieurs représentation de ces relations sont possibles, d'où l'existence de plusieurs catégories de modèles:

Les modèles de flux

Ces modèles, et les diagrammes associés sont destinés à représenter la circulation d'information, les flux de données, entre les différents acteurs du système.

Ces modèles, bien que ne fournissant qu'une représentation partielle du comportement de l'organisation, permettent d'obtenir une vue générale du système simple et compréhensible [GAN. 79], [FRE. 80].

Les modèles issus des réseaux de Pétri

Ces modèles sont basés sur les notions de condition et d'action: pour exécuter une action, il faut que certaines conditions soient remplies. L'exécution d'une action peut modifier l'environnement, et donc engendrer la vérification de nouvelles conditions.

De manière générale, on retrouve dans ces modèles le cycle "événement — point de synchronisation — action". Grâce à un formalisme assez simple, il est donc possible de représenter de manière très précise les condition d'enchaînement des traitements, donc le comportement du système [FIC. 88].

Les modèles "orientés objet"

Ces modèles sont également basés sur les événements. Seulement, dans ce cas, les événement sont répercutés sur les données (les objets) et seules les transitions (changements d'état d'objets) déclenchent des traitements [SHL. 88], [BLA. 90].

13.3. Démarche

Quelle que soit la méthode choisie, on recherche chaque fois à représenter de la manière la plus précise, la plus complète possible, les mêmes notions. Il est clair que selon le nombre de concepts qu'elle propose, et en fonction de leur facilité d'utilisation, une méthode permettra d'aboutir au résultat escompté plus facilement, plus surement qu'une autre.

L'idéal serait de pouvoir reprendre, dans chaque méthode, les modèles et les concepts intéressants, et de créer sa propre démarche exploitant ces modèles.

Cette vision des choses est à l'origine de travaux tel que ceux menés à propos de l'AD-CYCLE REPOSITORY [MAC. 91], d'outils "ouverts" tel que EXCELERATOR, mais également à l'origine de la prolifération "anarchique" des méthodes de conception.

Si certaines méthodes apporte plus de modèles, plus de concepts que d'autres, on a également pu observer que chaque méthode aborde le système d'information selon un angle spécifique, selon une démarche propre.

Dès lors, choisir une méthode n'est plus seulement une question du nombre de concepts et de modèles présents dans une méthode. Il s'agit avant tout d'adhérer à la vision du système d'information proposée par les concepteurs de la méthode. Cette adhésion sera guidée par des critères aussi divers que nombreux (perception personnelle du problème, type de système à réaliser, éducation, environnement,...).

13.4. In fine

Vu l'importance reconnue des phases de conception et de spécification, et le nombre croissant de méthodes de conception et de modèles associés, il nous semble important de souligner, à nouveau, l'importance et la nécessité de l'utilisation d'une méthode, quelle qu'elle soit.

Bibliographie

[BEN. 89] BENCI G., LINGAT J.-Y., RAMES J.-R., "Conception et prototypage des systèmes d'information: La méthode REMORA et ses outils logiciels Rubis et Rémograph", GENIE LOGICIEL & SYSTEMES EXPERTS, n°15, sept. 89.

[BLA.90] BLACKWELL A., OLSON L., "Understanding Network Management with OOA", IEEE Network Magazine, July 1990, pp. 23-28.

[BOD.89] BODART F., PIGNEUR Y., "CONCEPTION ASSISTEE DES SYSTEMES D'INFORMATION, Méthode, modèles et outils", MASSON, 2e éd., 1989.

[CAS. 87] CASTELANNI X., "Méthode générale d'analyse des applications informatiques", Masson, 1987.

[CHE. 76] CHEN P.P., "THE ENTITY-RELATIONSHIP MODEL: Toward a unified view of data", ACM Transaction on Data Base Systems, vol. 1, n°1, 1976.

[COD. 70] CODD E.F., "A relational model of data structures: a brief tutorial", ACM Sigfidet Workshop on "Data Description Access and Control", Californie, 1971.

[FIC. 88] FICHEFET J., "Introduction aux réseaux de PETRI", Notes de cours, Facultés Universitaires N.D. de la Paix, Institut d'Informatique, Namur, 1988.

[FOU. 82] FOUCAUT O., "MODELES ET OUTIL POUR LA CONCEPTION DES SYSTEMES D'INFORMATION DANS LES ORGANISATIONS, Projet REMORA", Université de NANCY I, U.E.R. de Sciences Mathématiques, Thèse de Doctorat, 11 juin 1982.

[FRE. 80] FREEMAN P., "REQUIREMENTS ANALYSIS AND SPECIFICATION : The first step", dans "Advances in Computer Technology - 1980", American Society of Mechanical Engineers, août 1980.

[FRI. 89] FRIAUD J.-M., "Les outils de conception des systèmes d'information", Direction des Etudes du CXP, Etude n°149, decembre 1989.

[GAN.79] GANE C., SARSON T., "STRUCTURED SYSTEMS ANALYSIS : Tools and techniques", Prentice-Hall, 1979.

[IGL. 84] INSTITUT DE GENIE LOGICIEL, "COURS DE LECTEUR SADT, Support de cours", Ref. 39-SADT, octobre 1984.

[IGL.83] INSTITUT DE GENIE LOGICIEL, "DE L'ANALYSE A LA CONCEPTION", ref. 50-SADT/MACH, juillet 1983.

[IEW] KNOWLEDGEWARE, "INFORMATION ENGINEERING WORKBENCH, Design Workstation User Guide".

[LIS. 90] LISSANDRE M.,"Maîtriser SADT", Armand Colin Ed., 1990.

[MAC. 91] MACIASZEK L. A., "AD/Cycle Repository Manager from Object-Oriented Perspective", ACM SIGSOFT Software Engineering Notes, vol. 16, n°1, 1991, p. 50.

[MAR. 78] de MARCO T., "Structured Analysis and System Specification", YOURDON PRESS, 1978.

[PEL. 86] PELLAUMAIL PH., "LA METHODE AXIAL : Conception d'un système d'information", Les Editions d'Organisation, 1986.

[QUA. 89] QUANG P.T., "MERISE/YOURDON", dans GENIE LOGICIEL & SYSTEMES EXPERTS, n°15, Septembre 1989.

[RIS. 89] de RISI L., "LA METHODE MEGA. Manuel de Référence", GAMMA INTERNATIONAL, janvier 1989.

[ROC.] ROCK-EVANS R., "Analysis within the Systems Development Life Cycle", Ed. Ovum Ltd, volume 4.

[ROC. 89] ROCK-EVANS R., ENGELIEN B., "ANALYSIS TECHNIQUES FOR CASE : a detailled Evaluation", Volume 1, Ed. Ovum Ltd, 1989

[ROL. 88] ROLLAND C., FOUCAUT O., BENCI G., "CONCEPTION DES SYSTEMES D'INFORMATION : La méthode remora", Eyrolles Ed., 1988.

[ROS. 77 a] ROSS D.T., SCHOMAN K.E. JR, "STRUCTURED ANALYSIS FOR REQUIREMENTS DEFINITION", dans I.E.E.E. Transactions on Software Engineering, Volume SE-3, Numéro 1, Janvier 1977, pp. 69 - 84.

[ROS. 77 b] ROSS D.T., "STRUCTURED ANALYSIS (SA) : A Language for communicating Ideas", dans I.E.E.E. TRANSACTIONS ON SOFTWARE ENGINEERING, Volume SE-3, Numéro 1, Janvier 1977, pp. 16 - 34.

[SHL.88] SHLAER S., MELLOR S.J., "OBJECT-ORIENTED SYSTEMS ANALYSIS - Modeling the World in Data", Yourdon Press, 1988.

[SOF. 79] SOFTECH INC., "An introduction to SADT", Ref. 9022-78R, 1979.

[SOF. 80] SOFTECH INC.,"SADT : Introduction", par MICHAEL F. CONNOR, Ref. 9595-7, 22 May 1980.

[TAB. 86] TABOURIER Y., "De l'Autre Côté de MERISE, Systèmes d'information et modèles d'entreprise", Les Editions d'Organisation, 1986.

[TAR. 89] TARDIEU H., A. ROCHFELD A., COLLETTI R., "LA METHODE MERISE: Tome 1 Principes et outils", Les Editions d'Organisation, 1989.

[THE. 89] THEYS M., "Comment intégrer les données & les traitements du modèle conceptuel d'un système informatique de gestion", Ecole de Commerce, ULB, dans "LE GENIE LOGICIEL ET SES APPLICATIONS", DEUXIEMES JOURNEES INTERNATIONALES - PROCEEDING / ACTES, Volume 1, Toulouse 1989.

[UNI. GM] UNISYS, "Guide méthodologique".

[YOU. 84] YOURDON E., CONSTANTINE L., "Structured Design", Yourdon Press, 1978, Proceeding of the International Conference on Data EngineerinG, IEEE Computer Society Press, 1984.