MASTER EN SCIENCES INFORMATIQUES

Étude d'une méthodologie et d'outils d'aide au développement d'applications de gestion

Delieux, Nathalie; Bénédicte, Delvaux

Award date:
1987

Awarding institution:
Universite de Namur

FACULTÉS
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

ETUDE D'UNE METHODOLOGIE ET D'OUTILS D'AIDE AU DEVELOPPEMENT D'APPLICATIONS DE GESTION

Nathalie DELIEUX

Bénédicte DELVAUX

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Promoteur : Mr. Baudouin LE CHARLIER.

Année académique 1986-1987.

REMERCIEMENTS

L'élaboration de ce mémoire a nécessité le concours de nombreuses personnes qui ont contribué à former une atmosphère propice à la recherche et à l'étude.

Nous tenons particulièrement à exprimer notre profonde gratitude à l'égard de Baudouin Le Charlier, promoteur de ce travail, pour les précieux conseils qu'ils nous a prodigués lors de fréquents entretiens ainsi que pour nous avoir offert son soutien moral, soutien indispensable à la réalisation d'un mémoire.

Nous adressons également nos plus vifs remerciements à Marie-Claire Verlaine, chef du service Méthodologie et Administration de l'Information à la Caisse Générale d'Epargne et de Retraite, pour nous avoir accordé de nombreuses entrevues qui ont permis un échange fructueux d'idées ainsi que pour nous avoir si aimablement intégrées à son équipe.

Nous éprouvons tout particulièrement de la reconnaissance envers Christian Graas, Willy Luyckx et Frans de Brabanter qui ont collaboré à ce travail. Ils nous ont consacré, avec beaucoup de dévouement, de nombreuses heures de leur temps précieux et ce, respectivement pour l'aspect méthodologie, l'aspect outils et l'aspect application.

Enfin, nous remercions tous les membres du service M.A.I. ainsi que de toutes les personnes de la CGER qui, de près ou de loin, ont contribué à la réalisation de notre projet. Leur accueil chaleureux et leur assistance quotidienne nous ont d'ailleurs permis d'approfondir notre analyse au-delà des limites initialement fixées. Toutes ces personnes nous ont par ailleurs laissé un excellent souvenir de notre première prise de contact avec un environnement professionnel.

TABLE DES MATIERES

CHAPITRE 2 : LES OUTILS D'AIDE A LA DOCUMENTATION ET AU DEVELOPPEMENT DE PROJETS INFORMATIQUES

CHAPITRE 3 : ELEMENTS DE SPECIFICATION D'OUTILS D'AIDE A LA DOCUMENTATION ET AU DEVELOPPEMENT SUR BASE DE L'APPLICATION "PUMA"

INTRODUCTION

Depuis de nombreuses années, maints efforts ont été effectués en vue de mettre au point des méthodologies pour le développement de projets informatiques. De nombreuses motivations sont à l'origine de ce mouvement, la principale étant la perspective de pouvoir augmenter la productivité des départements informatiques et ce, pour le développement de projets comme pour la phase de maintenance. Les autres motivations, sous-jacentes à la première sont d'améliorer la qualité des analyses et des programmes, d'assurer une meilleure correspondance du nouveau système aux besoins des utilisateurs et de faciliter la planification du développement.

En vue de répondre à ces objectifs, une méthodologie devra non seulement guider ses utilisateurs dans la conception et la maintenance d'un projet mais aussi fournir des indications quant à la gestion et au contrôle du projet (planification, encadrement budgétaire, rentabilité, adéquation entre les objectifs du projet et les objectifs à long terme de l'entreprise, exigence du projet au niveau de l'environnement technique,...).

Malheureusement, "la" méthodologie, panacée universelle, n'a pas encore vu le jour. Dans l'attente de sa découverte (éventuelle), nous pouvons faire le point sur la situation actuelle et tenter d'isoler quelques problèmes qui entravent le développement d'une méthodologie "idéale" et l'utilisation d'une méthodologie en particulier.

Ceci constituera le premier volet de notre étude où nous nous limiterons à l'aspect conception et maintenance de projets de type application de gestion. Néanmoins, nous soulignerons quelques problèmes relatifs à l'aspect gestion et contrôle d'un projet.

D'autres moyens sont également mis en oeuvre en vue d'augmenter la productivité des développeurs et d'améliorer la qualité de la documentation des projets; nous faisons allusion aux "outils", ou logiciels, d'aide au développement et à la documentation de projets informatiques. Etant donné le récent foisonnement de ce type de systèmes dans le monde informatique, il est intéressant de se demander, d'une part, dans quelle mesure ces outils peuvent répondre aux objectifs qui leur sont assignés et, d'autre part, quel rôle (éventuel) doit jouer la méthodologie dans l'installation et l'utilisation d'outils.

Le deuxième volet de notre étude a pour objectif de regrouper un ensemble de réflexions sur base desquelles nous pourrons énoncer des éléments de réponse à ces questions.

La démarche que nous avons suivie est en partie basée sur la pratique. En effet, après avoir eu durant nos études une vision plutôt théorique de l'utilisation de méthodologies et d'outils, nous avons eu l'occasion d'être confrontés au monde informatique lors de notre stage à la Caisse Générale d'Epargne et de Retraite.

Notre objectif étant de faire le point sur les problèmes dus à l'application d'une méthodologie, le meilleur moyen s'offrant à nous était de développer une application en vue d'éprouver la méthodologie qui nous était proposée.

En ce qui concerne les outils, nous avons étudié en particulier le dictionnaire des données en vue d'examiner comment nous aurions pu utiliser un tel outil lors du développement de notre application.
Remarquons dès à présent qu'il nous aurait été possible d'utiliser le dictionnaire des données en parallèle avec le développement de notre application mais ceci impliquait que nous nous conformions aux règles d'utilisation définies par la CGER. C'est pourquoi nous avons préféré adopter une démarche plus méthodique en étudiant tout d'abord les capacités intrinsèques du dictionnaire des données pour pouvoir définir par la suite un mode d'utilisation qui nous semble adéquat.

Nous insistons sur le fait que nous n'avons pas l'ambition de trouver des solutions universelles aux problèmes dus à l'utilisation d'une méthodologie et de certains outils mais plutôt de mentionner quelques réflexions qui pourraient guider les personnes confrontées à ces problèmes.

Dans une première partie, nous présenterons tous les concepts qui serviront de base à notre étude. Le premier chapitre sera relatif à l'aspect méthodologie. Nous présenterons tout d'abord la méthodologie qui nous a été proposée à la CGER que nous comparerons avec la méthodologie vue à l'Institut d'Informatique (F.Bodart et Y.Pigneur). Ensuite, nous observerons dans quelle mesure la méthodologie proposée à la CGER est utilisée dans la pratique. Le second chapitre concernera les outils d'aide au développement : il s'agit des dictionnaires des données que nous illustrerons par une description du dictionnaire utilisé à la CGER : DATAMANAGER, ainsi que d'autres outils dont nous donnerons un aperçu des tendances actuelles.

La seconde partie renfermera la partie essentielle de notre étude. Le premier chapitre reprendra une description de l'application que nous avons développée. Dans le second chapitre, nous exposerons les problèmes que nous avons rencontrés par rapport à la méthode qui nous était proposée ainsi que quelques éléments de réponse à ces problèmes. Dans le troisième chapitre, nous étudierons les fonctions que nous aurions pu attendre du dictionnaire DATAMANAGER pour documenter notre application; nous donnerons également les spécifications d'outils qui nous auraient été utiles.

La troisième partie fera état de conclusions générales.

Mais avant d'entamer la première partie, nous jugeons utile de présenter notre environnement de travail à la CGER.

En effet, en précisant, d'une part, la composition du département informatique et, d'autre part, les objectifs associés au service où nous avons été accueillies, nous situerons notre étude dans son contexte organisationnel.

Le service Méthodologie et Administration de l'Information (M.A.I.) s'intègre dans la structure du département informatique selon la figure 1 (p. 5).

Ce service a été créé en vue de proposer des méthodes pour le développement de projets et de remédier au manque de maîtrise de l'information au niveau du département informatique.
Son objectif est d'être un service de support et d'assistance dans le développement de projets informatiques; ses utilisateurs sont les autres services du CTI et les utilisateurs info-centre.

Le M.A.I. est composé de sept parties, appelées "sections", qui contribuent à la réalisation de cet objectif commun. La figure 2 (p. 6) est une représentation de ces différentes sections que nous décrivons ci-dessous :

1. Section administration de l'information

   Ses préoccupations sont :

   - le support au développement de projets par l'élaboration de méthodes et par une collaboration étroite avec les développeurs de projets;
   - le support méthodologique d'un point de vue théorique;
   - l'administration des données;
   - le contrôle des standards via des "naming conventions" pour les fichiers et les données de tous les projets de la CGER.

2. Section bases des données

   Cette section est responsable de deux systèmes : IMS et DB2.

   Ses préoccupations sont :

   - couvrir l'aspect strictement système (installation, system administration);
   - assurer l'administration des bases des données :
     - DB design (logique et physique),
     - suivi des applications,
     - manipulation des bases de données,
     - back up et réorganisations.

3. Section dictionnaire des données et outils de développement

   Le rôle de cette section est de soutenir et d'assurer le support au développement.

Les principaux outils sont :

- DATAMANAGER, un dictionnaire des données,
- PDF, destiné à supporter la méthode JSP (Jackson Structured Programming)
- TSO, un moniteur de télé-traitement,
- IMSADFII, un système de support pour le développement d'applications IMS,
- METACOBOL,
- outils divers (traitements de texte,...)

4. Info-centre

Cette section a pour objectif de fournir une aide aux utilisateurs (formation d'un User's Club). Cette assistance a trait aux manipulations de management (analyse financière, actuariat, gestion des portefeuilles, trésorerie,...) car les utilisateurs ont besoin d'extraire des informations et de faire des simulations.
Les outils utilisés doivent être simples et conviviaux.

5. Section Sécurité informatique

Ses préoccupations sont notamment :

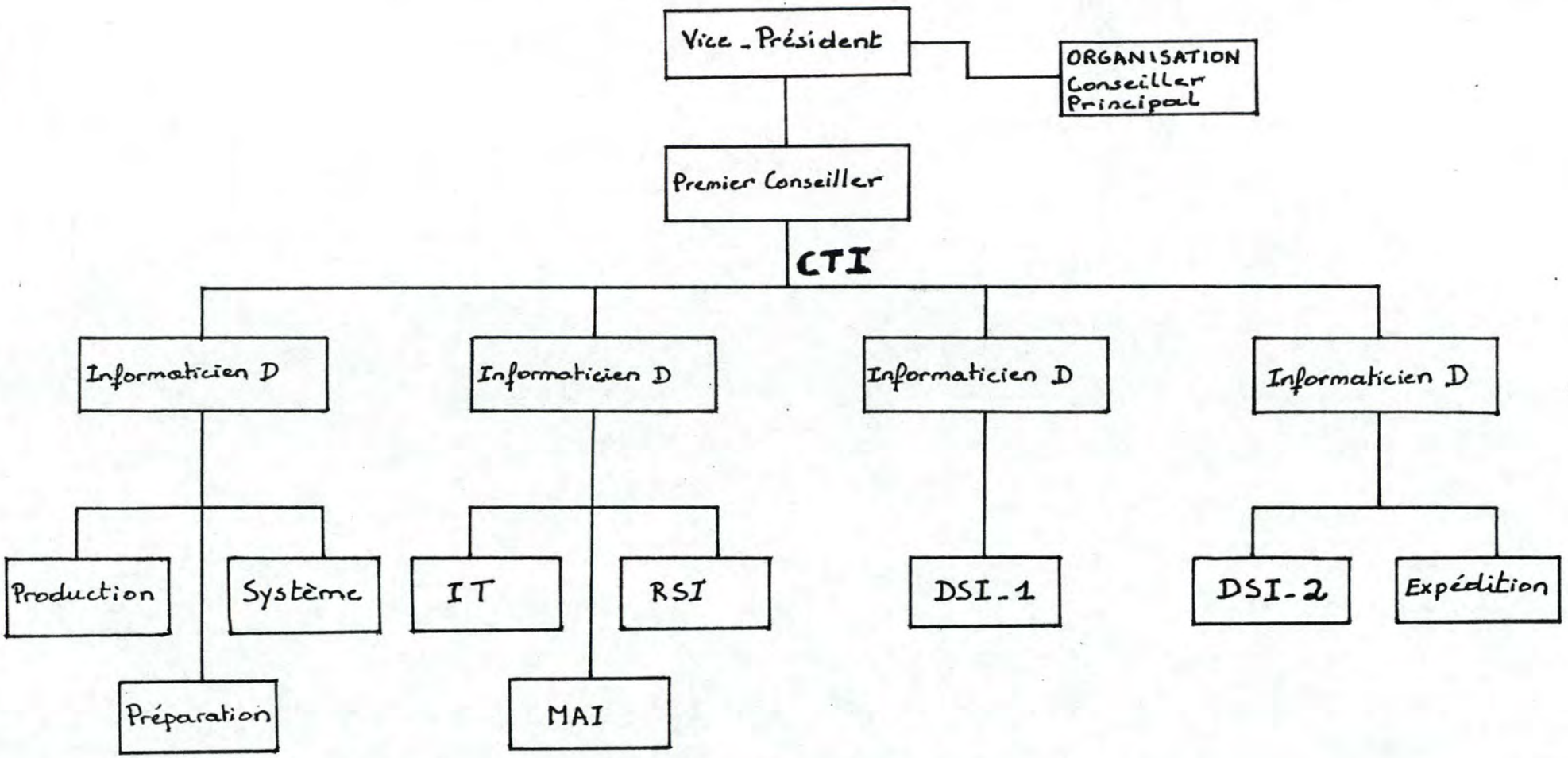
- les problèmes de confidentialité et de back up,
- le contrôle des badges à la salle des machines,
- l'aspect disaster-recovery,...

6. Formation

Cette section s'attache à la gestion des cours (formation pour l'utilisation de la méthodologie, des bases de données, des outils,...).
Dans la lignée d'un nouvel objectif de suivi de carrière, elle centralise les demandes du personnel CTI.

7. Secrétariat du CTI

fig.1 - SERVICES DE L'ORGANISATION ET DE L'INFORMATIQUE



CTI: Centre de Traitement de l'Information

IT : Installation Technique

RSI : Recherche et Support Informatique

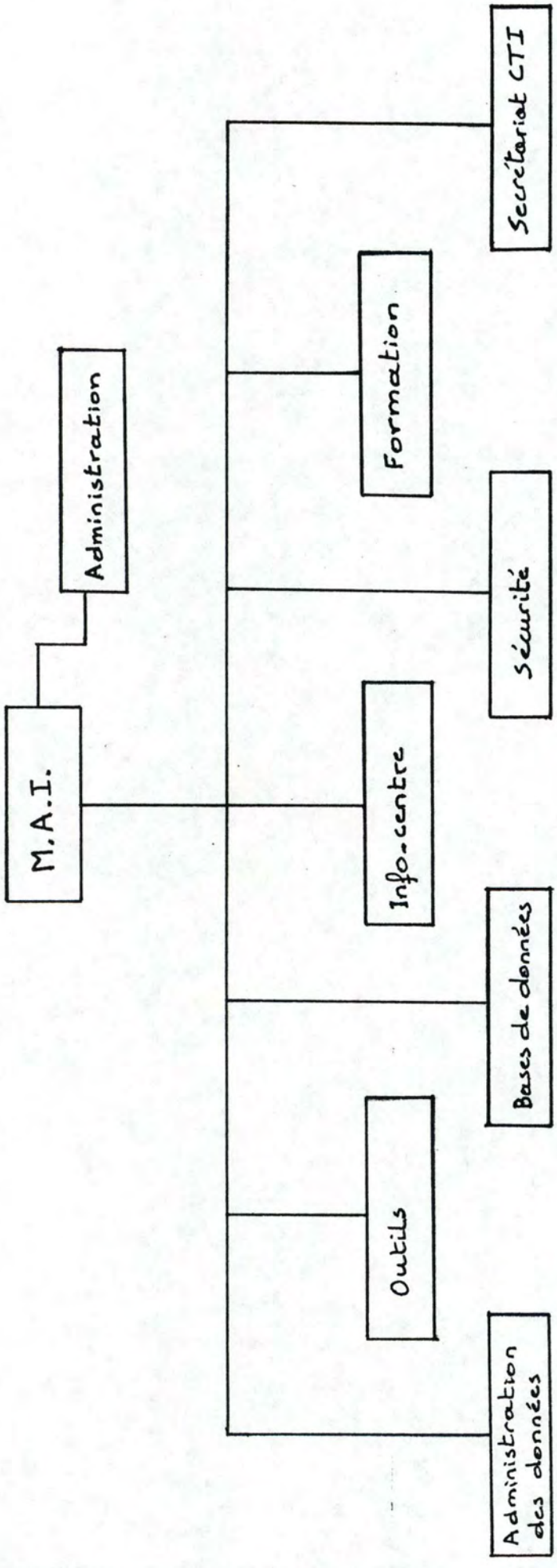
MAI : Méthodologie et Administration de l'Information

DSI_1 : Développement de Systèmes Informatiques _entité 1 (banque)

DSI_2 : Développement de Systèmes Informatiques _entité 2 (assurances, gestion du personnel)

fig. 2.

METHODOLOGIE ET ADMINISTRATION DE L'INFORMATION - M.A.I.



PREMIERE PARTIE

CONCEPTS DE BASE : L'ASPECT METHODOLOGIE ET L'ASPECT DES OUTILS

CHAPITRE 1

L'ASPECT METHODOLOGIE D'AIDE AU DEVELOPPEMENT D'APPLICATIONS DE GESTION

1. INTRODUCTION

Dans ce chapitre, nous présenterons différents aspects méthodologiques qui serviront de base à notre étude.

Nous décrirons tout d'abord la méthodologie que le service M.A.I. proposait au DSI lorsque nous avons commencé notre stage à la CGER. Ensuite, nous la comparerons d'un point de vue strictement théorique avec la méthodologie que nous avons étudiée à l'Institut d'Informatique (méthodologie proposée par François Bodart et Yves Pigneur). Mais, après avoir constaté que la méthodologie proposée à la CGER n'était pas appliquée à la lettre dans les départements informatiques, nous avons voulu observer la manière dont certains projets étaient développés. Nous exposerons donc un compte rendu de ces observations, accompagné de quelques réflexions.
En dernier lieu, nous dirons quelques mots au sujet de la stratégie que compte utiliser le M.A.I. en vue d'imposer au DSI une démarche plus méthodique que la démarche observée.

Par souci d'indépendance vis-à-vis des moyens mis en oeuvre dans l'utilisation d'une méthodologie, nous avons volontairement fait abstraction des outils proposés comme support du développement de projets informatiques. Certains outils seront examinés en détail dans le second chapitre de cette même partie.

Dans ce cadre, nous adopterons une démarche neutre dans la mesure où nous nous en tiendrons à une description des faits tels que nous les avons perçus grâce aux divers documents consultés et aux nombreux entretiens que nous avons eus avec les responsables des méthodes.
Nous n'avons donc pas l'intention de porter un jugement de valeur sur le contenu des points qui vont suivre. Néanmoins, certains sujets abordés nous ont incités à en donner les points forts et les points faibles. Ceci aura pour effet d'annoncer quelques sujets de réflexion auxquels nous ferons référence dans la suite de notre étude.

2. DESCRIPTION DE LA METHODOLOGIE PROPOSEE A LA CGER

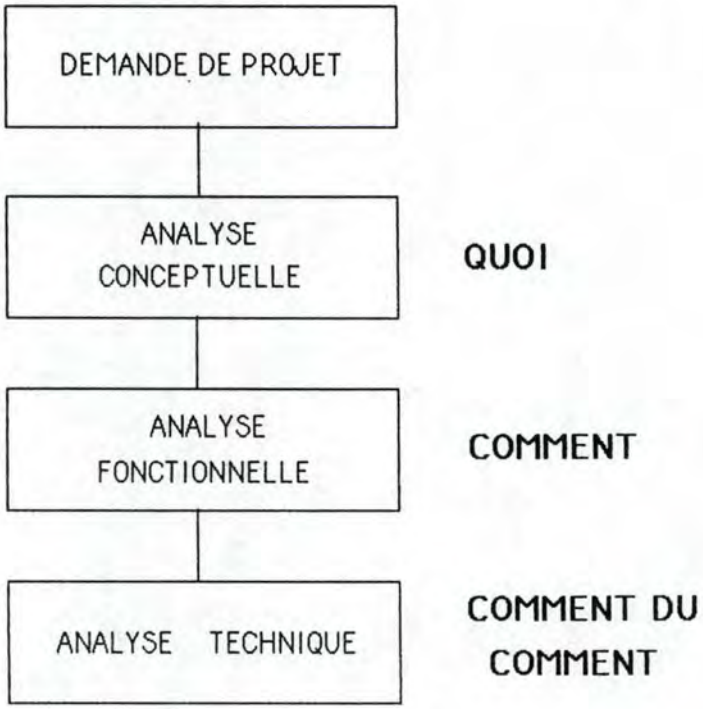
2.1. INTRODUCTION

"Le but de la méthodologie est de répondre de manière efficiente aux desiderata de l'utilisateur. On utilisera un mode uniforme de raisonnement qui devrait permettre l'interchangeabilité et la formation des personnes et permettre de découvrir au plus tôt les erreurs et les oublis dans la définition du problème.

L'utilisateur, quant à lui, doit être impliqué dans le processus de développement, ce qui nécessite une amélioration de la communication entre les concepteurs et lui.

En vue d'obtenir des spécifications complètes, exactes et non ambiguës, il sera nécessaire de parler un langage commun, de suivre des normes à travers une méthode." [GRAAS (a), pp 3-6]

Cette méthode se base sur la découpe du cycle de vie du développement en un certain nombre de phases bien définies.
Elle correspond donc au schéma type adopté par la plupart des méthodologies :



La démarche méthodologique décrite ici couvre toutes les phases du développement (de la définition de projet à la mise en production) ainsi que quelques opérations à caractère administratif, à savoir, la demande de développement et la demande de code d'application.

La description de cette méthodologie résulte de l'analyse des documents qui nous ont été fournis par les responsables des méthodes du service MAI, ainsi que des nombreux entretiens qu'ils nous ont accordés.

Dans un premier temps, nous détaillerons les opérations à caractère administratif, ensuite, nous décrirons les différentes phases du développement d'un projet informatique.
De temps à autre, il sera fait mention de certains services de la CGER, à ce sujet, nous invitons le lecteur à se référer à l'organigramme de l'entreprise présenté dans l'introduction générale.

2.2. DEMARCHES A CARACTERE ADMINISTRATIF

2.2.1. LA DEMANDE DE DEVELOPPEMENT

Tout développement de projet doit être précédé d'une demande de développement dont le but est d'obtenir le feu vert pour le démarrage du projet.

La demande de développement est un formulaire à remplir lors de la demande d'un nouveau système, d'une modification ou extension conceptuelle d'un système existant. Pour mémoire, c'est ce même formulaire qui devra être rempli lors de la demande d'un nouveau matériel, voir annexe A1. [MAI (c), p.2]

Cette demande doit être complétée sous la responsabilité de l'utilisateur (i.e. le responsable de la définition et de l'exécution des traitements du système [MAI (f), p.1]) en collaboration avec le CTI.

Le document contiendra, outre des informations générales (le groupe de travail, les personnes et services concernés par le nouveau projet, code d'imputation du budget, etc), la description du contexte actuel, les causes et origines du problème, les avantages attendus ainsi que les contraintes de réalisation (aussi bien temporelles que financières) du nouveau système.

Une fois la demande approuvée par le DSI, elle doit être transmise pour information au M.A.I.

2.2.2. LA DEMANDE DE CODE D'APPLICATION

Cette opération ne fait pas partie intégrante de la méthodologie, de ce fait, elle est indépendante, dans une certaine mesure, des autres étapes de la démarche. Néanmoins, nous avons choisi de la mentionner ici car elle présente un intérêt pratique du point de vue de l'administration du projet.

Dans ce point, nous levons l'hypothèse d'indépendance vis-à-vis des outils utilisés, en l'occurrence le dictionnaire des données, en vue de pouvoir citer une des fonctions du code d'application.
Un nouveau code d'application est attribué à un système s'il forme un ensemble homogène suffisamment indépendant des projets existants. Si tel n'est pas le cas, le système se verra attribuer le code d'application de la classe des projets présentant des points de similitude avec lui, plusieurs projets peuvent en effet avoir le même code d'application.

"Une application est définie (selon B. Brems) comme étant un ensemble de programmes, de procédures, de fichiers,... ayant trait à un ensemble de données et nécessaire à la satisfaction des besoins de l'utilisateur. Les systèmes suivants correspondent à la définition d'une application : comptes universels, crédits à court terme, comptes d'épargne mais aussi IMS, info-centre,..." [GRAAS (b), p.11]

C'est au DSI à établir, par le formulaire ad hoc, la demande du code d'application qui devra être approuvée par le "development control" qui attribue le code d'application (voir commentaires de l'annexe A2) et par le M.A.I.

Ce code a pour fonction de classer le projet au sein du dictionnaire des données et de permettre au service sécurité (au M.A.I) de contrôler les accès aux ressources informatiques (fichiers,transactions,...) relatives au projet.
Le code doit être attribué avant la phase de l'analyse conceptuelle si le projet doit être documenté par le dictionnaire des données, dans le cas contraire il peut être attribué juste avant la phase de programmation (car ce code permet l'attribution, notamment, des disques pour le système).

2.3. PHASES DU DEVELOPPEMENT DE PROJET

2.3.1. LA DEFINITION DE PROJET

La définition de projet précise les frontières, les contraintes, les objectifs et les besoins relatifs au projet à développer et ce, en fonction du système existant.
Le dossier à remplir comporte trois volets : l'étude de l'existant, les objectifs et contraintes du nouveau système et la liste des solutions possibles.

Une fois ce dossier établi par les membres du groupe de travail (utilisateurs et responsables du développement du projet), une instance de décision, dont l'importance varie suivant l'ampleur du projet, disposera de toutes les informations nécessaires sur lesquelles elle se basera pour effectuer un choix parmi les différentes solutions proposées.

a) L'étude de l'existant

Elle résulte d'interviews et d'études de documents réalisées par les membres du groupe de travail assigné au projet. Elle comporte trois volets : le recensement des activités et tâches de l'existant, celui des documents et des fichiers actuels ainsi que la représentation schématique du système existant.

En ce qui concerne les traitements, chaque activité et tâche est décrite par une fiche reprenant les informations suivantes : identification, événement déclenchant, informations en entrée, en sortie, mises à jour, description et commentaires éventuels.

"Par tâche, on entend un ensemble organisé d'opérations élémentaires, perçues et désignées globalement, manuelles et/ou automatisées. Ces opérations sont répétitives pour chaque échange d'informations transformant selon des règles prédéfinies des informations d'entrée en informations de sortie. Une tâche est, par ailleurs, caractérisée par son unicité de temps, de lieu et d'action." [GRAAS (a), p.26]

La signification à attacher à ces trois types d'unicités a été précisée par IBM [ibidem] comme suit :
- un traitement répond à une condition d'unicité dans le temps s'il effectue une séquence unique d'opérations sans interruption, séquence répétitive pour chaque échange d'informations;
- l'unicité de lieu implique que cette séquence est exécutée à un poste de travail;
- l'unicité d'action caractérise un traitement qui utilise des règles fixes de transformation de l'information.

"Par activité, on entend un ensemble cohérent de tâches nécessaires à l'accomplissement d'une finalité permanente et essentielle du système." [GRAAS (b), p.3]

En ce qui concerne les informations, les documents, fichiers et écrans sont également définis par une fiche reprenant les informations suivantes : identification, description, caractéristiques et commentaires éventuels. Chaque description doit être accompagnée d'un exemple (document vierge ou rempli, description de la structure du fichier, copie de l'écran,...).

La représentation schématique du système actuel se fait sur base de modèles. Les modèles proposés sont de deux types : un diagramme général pour l'ensemble du système et un diagramme détaillé pour les cas plus complexes.

Le diagramme général met en relation les tâches ou activités avec leur exécutant, de plus, il indique de quelle manière se succèdent les différents traitements ainsi que leur durée (si possible). Peuvent être considérés comme exécutants des personnes à titre individuel, des groupes de personnes mais aussi des machines, etc. Voir annexe A3.

Le diagramme détaillé consiste en une ventilation du diagramme général. En effet, il est enrichi par des indications sur les flux d'informations ainsi que par la possibilité d'introduire des structures représentant des objets physiques, des points de décision,...Voir annexe A4.
Une amélioration de ce diagramme est à l'étude actuellement.

Suite à cette analyse, il sera procédé à la critique du système existant où l'on dégagera les anomalies, les disfonctionnements, etc.

b) Objectifs et contraintes du nouveau système

En ce qui concerne les objectifs, on distingue les objectifs stratégiques et les besoins exprimés par l'utilisateur.
Les objectifs doivent répondre aux anomalies du système existant; si tel n'est pas le cas, une explication est souhaitée.
Les besoins principaux exprimés par l'utilisateur seront rédigés en texte libre.
Les contraintes sont temporelles, financières, légales, réglementaires ou définies en fonction de l'environnement (sécurité, temps de réponse, communication avec d'autres applications,...)

c) Solutions possibles

Il est souhaitable d'envisager plusieurs solutions possibles à soumettre à une instance de décision.

Pour chacune des solutions, on fournira une description précise, un modèle (en l'occurrence le diagramme détaillé décrit dans l'étude de l'existant), une analyse des coûts ainsi qu'un planning recouvrant toutes les phases de développement jusqu'au passage en production.

Remarque : La définition de projet étant un document fastidieux à compléter, le service M.A.I. étudie comment la simplifier, en particulier au niveau de l'étude de l'existant et au niveau des modèles de circulation des informations utilisés. Les responsables des méthodes proposent de s'orienter vers la production d'un diagramme de flux de données comme "output" de la définition de projet, avec délimitation des systèmes actuels et futurs ainsi que des phases de réalisation.

2.3.2. L'ANALYSE CONCEPTUELLE

"Cette analyse résulte d'une confrontation entre, d'une part, l'utilisateur et, d'autre part, le responsable du projet informatique sous-jacent (du DSI) et ce, avec le support du M.A.I.
Les réunions se feront en présence d'un rapporteur qui sera chargé d'acter les décisions prises et de noter les problèmes en suspens." [MAI (f), p.4]

L'analyse conceptuelle consiste en l'élaboration de modèles du réel perçu, modèles qui respectent une contrainte d'invariance. En effet, ils expriment la sémantique de la structuration des informations à l'exclusion de toute contingence technique ou organisationnelle. Ces modèles exigent de la part de leur auteur qu'il se place à un niveau d'abstraction suffisant pour rester indépendant de tout choix d'implémentation.

L'analyse conceptuelle doit pouvoir assurer un maximum de stabilité des bases du nouveau système afin d'en permettre, le plus longtemps possible, une maintenance aisée. "Cependant, on notera toutes les contraintes déjà connues pouvant influencer ultérieurement la conception, le développement ou le fonctionnement du système, afin d'en tenir compte en temps opportun." [MAI (a), p.2]
Ceci nous amène à introduire une nouvelle filière parallèle à l'analyse et qui sera présente jusqu'à la fin du développement d'un projet. Il s'agit de la filière de spécification des caractéristiques fonctionnelles du système. Ces contraintes guideront les choix ultérieurs et serviront entre autres de critères de qualité lors des tests d'application.

L'analyse conceptuelle comporte deux volets : l'analyse des données et l'analyse des traitements, analyses qui ne sont pas indépendantes.

a) L'analyse conceptuelle des données

A l'issue de cette analyse, un modèle conceptuel des données sera fourni. Celui-ci, s'il est approuvé par le M.A.I., constituera une référence pour la construction d'architectures de supports d'information dans les étapes ultérieures.

L'approche qui a été choisie à la CGER est celle du modèle entité/relation, "étant donné que ce modèle est très répandu et qu'il est basé sur un petit nombre de concepts précis qui offrent une grande souplesse d'adaptation aux cas réels. Il constitue un excellent moyen d'aborder un problème, ainsi qu'un outil privilégié de dialogue entre utilisateurs et informaticiens." [ibidem, p.3]

Nous reprendrons ici la description des concepts du modèle entité/relation. Ensuite, nous indiquerons pour chaque concept la représentation graphique proposée.

Description des concepts du modèle entité/relation

Dans la documentation de la CGER [MAI (a)], ce modèle est présenté de manière pédagogique. En effet, une méthode de construction du schéma est proposée, elle détaille chacune des étapes à réaliser en définissant les concepts y afférents et en les illustrant par un exemple suivi à travers tout le processus.

Ces concepts seront décrits au travers des étapes suivantes :

- Etablissement de la liste brute des informations du système à modéliser

  Cette liste est établie sur base de l'étude de l'existant ainsi que sur base des discussions avec les utilisateurs.

- Epuration de la liste brute des informations

  Grâce à l'étude de la sémantique des informations, les répétitions, les synonymes, les homonymes seront éliminés.
  Les constituants de cette liste d'informations sont des propriétés d'objets. Ces propriétés sont soit des éléments, soit des agrégats.
  Un élément est un atome d'information qui perdrait toute sa signification si on tentait de le scinder.

Un agrégat est un ensemble de données, considérées comme un tout d'un point de vue logique. Il est composé d'autres éléments ou d'autres agrégats.

- Dégagement des types d'entité

Par simplification d'écriture, on utilisera par la suite "entité" pour "type d'entité" et "relation" pour "type de relation".

Une entité est un objet présentant un intérêt pour l'entreprise, possédant une existance propre et dont chaque occurrence peut être identifiée de façon univoque par la valeur prise par une de ses propriétés appelée identifiant.

- Rattacher leurs propriétés (ou attributs) aux entités

- Définir les relations entre entités

Une relation est un objet définissant une correspondance entre deux ou plusieurs entités.

Une relation n'a pas d'existence propre; cette dernière dépend de celle des entités associées.

L'identifiant d'une relation est formé de la concaténation des identifiants des entités qu'elle relie. Le modèle ne prévoit apparemment pas qu'une relation puisse avoir un identifiant propre.

Remarques:
- Plusieurs relations peuvent exister entre les mêmes entités.
- Une relation peut être réflexive et relier une entité à elle-même.

- Rattacher les propriétés restantes aux relations

Une relation peut, en effet, être porteuse ou non d'informations.

- Déterminer les cardinalités de chaque couple entité/relation

Les cardinalités expriment la participation à un type de relation des types d'entités qui la composent. Il s'agit des nombres minimum et maximum d'occurrences de la relation pouvant exister pour une occurrence de l'entité considérée.

Les cardinalités sont importantes car elles expriment des règles de gestion. Par exemple, supposons une cardinalité (1,*), qui exprime une contrainte d'existence, entre un type d'entité (T.E.) et un type de relation (T.R.); ceci signifie que l'on décide

qu'une entité de type T.E. ne sera pas enregistrée dans le système si elle ne participe pas à une relation du type T.R.

Dans le cas des relations réflexives, il est utile d'indiquer les rôles joués par les relations, conjointement avec les cardinalités correspondantes.

- Simplifier éventuellement le modèle avec des contraintes d'intégrité

Il s'agit de voir si notamment on ne peut pas remplacer des relations d'ordre n (définies sur n entités) par des relations binaires (définies sur 2 entités non nécessairement distinctes) en se servant des règles de gestion de l'entreprise.

Nous devons mentionner que la documentation consultée indique que ce processus est itératif. Cependant, il n'est pas précisé sur quoi peut porter cette itération. Pour notre part, nous pensons qu'elle porte sur la mise à jour du schéma conceptuel et ce, chaque fois qu'une nouvelle information est ajoutée à la liste brute des informations et/ou quand les membres du groupe de travail hésitent entre plusieurs schémas entité/relation pour un même problème.

Représentation graphique

"En l'absence d'outil graphique, le symbolisme qui suit est simplement conseillé. Le groupe de travail peut donc choisir une autre représentation à condition de préciser laquelle et de l'utiliser tout au long de l'étude." [MAI (a), p.13]

- Entité et identifiant

L'entité est représentée par un rectangle contenant dans un cartouche supérieur le nom du type d'entité; le rectangle contient l'identifiant souligné.

- Relation

La relation est représentée par un ovale relié aux entités associées et dans lequel on indique le nom de la relation.

- Propriétés d'une entité (d'une relation)

La liste des propriétés d'une entité (d'une relation) est mentionnée dans le rectangle (ovale) ou en annexe selon la taille.

- Cardinalité

La cardinalité est représentée par un couple d'entiers (min,max) noté le long de la ligne reliant l'entité et la relation.

En conclusion, nous pouvons mentionner que la méthode proposée est une méthode 'bottom-up'. Pour notre part, il nous semble qu'une méthode 'top-down' serait plus proche du processus de modélisation.
Par ailleurs, la terminologie utilisée pourrait prêter à confusion (entité pour type d'entité ou pour occurrence d'un type d'entité), elle pourrait de ce fait induire en erreur une personne qui ne connaîtrait pas le modèle.

b) L'analyse conceptuelle des traitements

Cette analyse fait appel à plusieurs modèles : un modèle de découpe que nous appellerons modèle de décomposition des traitements, un modèle de description des traitements et un modèle de la dynamique.

- Le modèle de décomposition des traitements

Il constitue une approche "top-down" du problème, c'est-à-dire que l'on "réduit un problème global en une série de sous-problèmes moins complexes. Le résultat obtenu est une arborescence où tout élément de niveau intermédiaire i (i>1) provient de la décomposition d'un seul élément de niveau i-1 et se décompose en n (n>=1) éléments de niveau i+1.
Chaque élément correspondra à un traitement bien déterminé et la sémantique associée à ce modèle est qu'un traitement de niveau i+1 "fait partie" d'un traitement de niveau i." [BODART (a), p.51]

Etant donné que chaque niveau ou groupe de niveau a un intérêt propre selon son degré de détail, il est utile, par souci de standardisation, de pouvoir les distinguer. Ainsi, on nommera ces différents groupes : système, activité et tâche.

"Un système est un ensemble de traitements et de données fortement liés entre eux par des relations internes au système et pouvant être séparés de l'environnement du système tout en possédant certaines relations avec cet environnement." [GRAAS (b), p.22b]

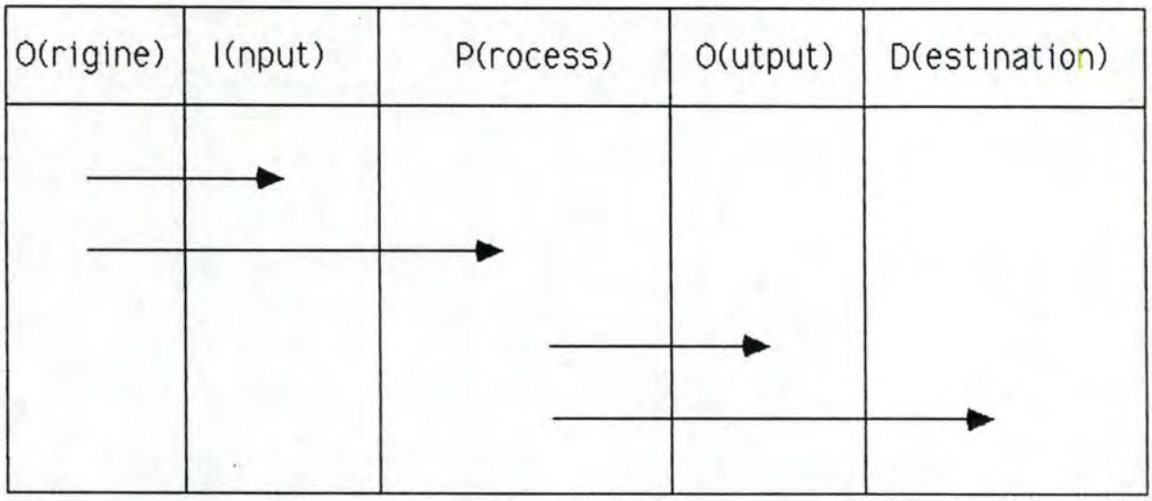
Les concepts d'activité et tâche ont été définis dans l'étude de l'existant.

Sur base de cette nomenclature, le modèle peut être précisé : le traitement de niveau 1 répondra aux caractéristiques d'un système, la découpe par raffinements successifs portera sur des traitements de type activité et les éléments terminaux seront de type tâche. L'arborescence comporte donc plusieurs niveaux

de type activité d'où la définition assez large de ce traitement. Ce modèle peut également être représenté par un Schéma Hiérarchique des Activités et Tâches (SHAT). Voir annexe A5.

- Le modèle de description d'une tâche

Une fois la découpe des traitements achevée, les tâches peuvent être décrites via un tableau HIPO (Hierarchical Input Process Output). Ce tableau se présente comme suit :



- la colonne I(nput) contient le nom des éléments utilisés en entrée par la tâche. Ces éléments sont soit des entités (au sens du schéma entité/relation), soit des informations circulant de tâche en tâche, ayant souvent une existence temporaire et pour lesquelles aucune décision de matérialisation n'a été prise, soit encore des documents vecteurs d'information.
- la colonne O(rigine) reprend l'origine de chacun de ces éléments,
- la colonne O(utput) renferme les éléments qui sont produits ou traités par la tâche,
- la colonne D(estination) indique la destination de ces éléments,
- la partie P(rocess) reprend une description du traitement associé à la tâche.

Ce tableau sera accompagné d'un texte précisant l'objectif de la tâche, les événements initiateurs et induits plus, éventuellement, quelques remarques.

- Le modèle dynamique

Ce modèle a été introduit pour servir de complément au modèle de décomposition des traitements quand la découpe résultant de ce dernier est considérée comme étant trop stricte pour répondre à certains besoins.

Le modèle dynamique se base sur le concept de "process"; un process étant "un ensemble de décisions et d'activités et tâches y relatives nécessaires à la réalisation dans le temps d'un traitement donné, faisant appel à 0,1 ou plusieurs entités." [GRAAS (b), p.30]
"Un process exprime un enchaînement de plusieurs tâches faisant partie éventuellement d'activités différentes, enchaînement qui correspond au problème de base de l'utilisateur." [ibidem, p.33] Voir annexe A6.

- La consolidation

Une fois l'arborescence obtenue, il est nécessaire de contrôler la cohérence de la découpe et la complétude du système, cela consiste à procéder à une consolidation du travail effectué.

Les moyens qui permettent ces deux types de contrôle sont les diagrammes tâches-utilisateurs (DTU) et les diagrammes tâches-entités (DTE).

Le DTU est le diagramme détaillé tâches-utilisateurs tel qu'il a été présenté dans le point consacré à l'étude de l'existant.
Nous n'avons pas trouvé d'indications quant à la manière dont ce diagramme peut être exploité en vue de procéder à une consolidation.

Le DTE est un diagramme ayant pour ordonnée les tâches et pour abscisse les entités. Un point d'intersection indique si une entité est ou non utilisée pour une tâche et, si oui, si cette donnée est utilisée en entrée, en sortie ou en mise à jour. Voir annexe A7.

Remarque : le DTE n'est pas un diagramme très clair car les documents que nous avons consultés ne précisent pas ce qu'il faut entendre par "entité".
Par conséquent, sa contribution à l'étape de consolidation ne paraît pas évidente, ce qui explique qu'il est souvent laissé de côté.
Le M.A.I. étudie d'autres moyens visant à contrôler la cohérence de la découpe.

2.3.3. L'ANALYSE FONCTIONNELLE

"Cette phase a pour but de fournir l'information nécessaire à la construction de l'architecture des programmes et de fournir au responsable bases des

données un modèle logique des données qui lui permettra de construire la base des données physique. A l'issue de cette phase, aura été précisée de façon claire et complète la manière dont l'utilisateur veut travailler avec son système." [MAI (b), p.3]

Les acteurs sont :
- du côté du développement, le responsable SGBD et l'équipe informatique qui a participé à la première phase, à qui l'on a joint un analyste;
- du côté des utilisateurs, les responsables de la définition et de l'exécution des traitements ainsi que des représentants du personnel qui exécutera réellement les opérations.

Un rapporteur sera toujours présent dans cette phase.

Le travail effectué en vue d'atteindre ces objectifs se base sur les différents modèles élaborés dans l'analyse conceptuelle : le modèle conceptuel des données, le modèle de décomposition des traitements et, éventuellement, le modèle de la dynamique.

De la même manière que pour l'analyse conceptuelle, on distinguera de nouveau l'analyse des données et l'analyse des traitements, quoique leur interaction soit plus importante ici.
Ne sera également pas négligée la spécification des caractéristiques fonctionnelles introduites à partir de la phase de l'analyse conceptuelle.

a) L'analyse fonctionnelle des traitements

Cette analyse se déroule à deux niveaux : celui des traitements interactifs et celui des traitements batch.

En ce qui concerne les tâches interactives, l'analyse fonctionnelle consiste à décrire pour chacune d'entre elles le dialogue utilisateur/système, c'est-à-dire un scénario élaboré sur base d'actions effectuées, d'une part, par l'utilisateur et, d'autre part, par la tâche (côté système). On procèdera ensuite à la spécification des écrans supportant le dialogue.
Ces descriptions peuvent aussi porter sur deux nouvelles conceptions d'un traitement : la transaction et la fonction.

"Une transaction utilisateur est une opération élémentaire, composante d'une ou plusieurs tâches et qui constitue une unité d'interaction entre l'utilisateur et le système d'information." [GRAAS (a), p.46]

Une transaction base de données est une opération élémentaire, composante d'une ou plusieurs tâches et qui constitue une unité d'interaction entre le système et la base de données.

"Une fonction est une composante d'une ou plusieurs tâches et/ou transactions exécutant de manière immuable et indépendante de l'environnement une séquence d'opérations élémentaires formant un tout logique, 'fonctionnellement réentrant'." [ibidem]
Notons que la fonction peut également interagir avec l'utilisateur.

La différence entre la transaction base de données et la fonction réside dans le fait qu'une transaction fait toujours au moins un accès ponctuel à la base des données, la fonction, jamais.

A ce sujet, il faut rappeler que la méthodologie est orientée données; de ce fait, sa philosophie est ici d'isoler les traitements qui interagissent avec la base des données de ceux qui en restent indépendants. Cette distinction permet de réduire le coût des modifications en cas de changement de SGBD.

"En ce qui concerne les tâches batch, on définira la logique de l'enchaînement des traitements ainsi que les états successifs." [MAI (b), p.3]

"Après l'analyse, on associera à chaque tâche interactive ou batch des paramètres qualitatifs et quantitatifs (on-line ou batch, priorité, périodicité, temps de réponse,...)" [GRAAS (a), p.42]

b) L'analyse fonctionnelle des données

Dans un premier temps, il sera procédé à la normalisation du schéma conceptuel des données.

"Le but de la normalisation est de parvenir par raffinements successifs à des ensembles de données possédant les meilleures propriétés de mise à jour.
Sa mise en oeuvre permet l'évolution de la maintenance des bases de données sans influencer la structure logique des données et les programmes d'application. (...). Cette démarche permet également de valider le modèle conceptuel des données en s'assurant que les informations y ont été regroupées de manière cohérente au sein des objets." [MAI (e), p.1]

"La normalisation est principalement réalisée selon les trois premières règles :
1. Une relation est en première forme normale (F.N.) si aucun attribut n'a de valeurs multiples pour une valeur donnée de la clé primaire.
2. Une relation est en deuxième F.N. si elle est en première F.N. et si tous les attributs autres que la clé primaire dépendent de la totalité de cette clé.
3. Une relation est en troisième F.N. si elle est en deuxième F.N. et si elle ne contient pas de dépendance transitive." [GRAAS (b), p.51] Voir annexe A8.

Remarquons que si le modèle conceptuel est soigné, il est souvent en deuxième forme normale.

C'est de façon totalement arbitraire que nous avons classé la normalisation dans la phase fonctionnelle . Elle pourrait tout aussi bien être réalisée à la fin de la phase conceptuelle.

Le but de l'analyse fonctionnelle des données est de transformer le modèle conceptuel des données en un modèle logique des données qui tient compte des accès.
Cette phase est à l'étude pour le moment (cfr. PALOMA décrit au point 6).
Nous citerons néanmoins la démarche adoptée jusqu'à présent.

Tout d'abord, on procèdera à la quantification statique des volumes des différentes entités et relations du schéma conceptuel des données (en terme de nombre d'occurrences), ce qui sera repris dans un modèle statique (M.S.).Voir annexe A9.

Ensuite, on analysera l'accès aux données à l'aide de la partie système du dialogue élaboré lors de l'analyse fonctionnelle des traitements.
Cette analyse se fera au niveau des principales tâches (batches importants ou tâches interactives fréquentes).
Ainsi, pour chacune de ces tâches, on réalisera un sous-schéma des données reprenant les chemins d'accès, leur fréquence, ainsi que leurs caractéristiques. Ces informations seront reprises dans un modèle dynamique (M.D.). Voir annexe A10.

Sur base de cette analyse, on peut attribuer un poids à chaque tâche représentant son coût du point de vue des accès.

Finalement, les sous-schémas obtenus peuvent être intégrés en un schéma logique global des accès qui représente la manière dont les données seront logiquement accédées.

Remarque : nous n'avons pas trouvé d'informations quant à la démarche à adopter pour intégrer les sous-schémas.

2.3.4. L'ANALYSE TECHNIQUE

Le modèle logique élaboré lors de l'analyse fonctionnelle est traduit en un modèle physique bien structuré, performant et respectant les contraintes du SGBD utilisé.

Ce travail est effectué par les auteurs du modèle des données en collaboration avec le groupe DBA (Data Base Administration) et tiendra compte des volumes et des fréquences d'accès dégagés dans le phase précédente.

Du côté des traitements, les analystes procèderont à une description de la structure et du mode d'enchaînement des applications et des programmes.

2.3.5. LA PROGRAMMATION

Pour la programmation, la méthode de Jackson peut être utilisée (JSP : Jackson Structured Programming). Cette méthode pourrait être appliquée aux tâches issues de l'analyse conceptuelle, tâches significatives et bien adaptées à ce mode de raisonnement.
Dans ce cas, la première étape de la méthode JSP est déjà réalisée, à savoir "l'étape de spécification où le problème est exposé clairement et où les entrées/sorties sont décrites." [VAN'T DACK, p.2]

Les étapes suivantes sont :

- "l'étape données
  Les structures de données nécessaires sont définies sur base des spécifications de la première étape;

- l'étape programme
  A partir des structures de données, on déduit la structure du programme à l'aide des correspondances existant entre les composants des différentes structures de données;

- l'étape opérations
  A ce niveau, on introduit dans la structure des opérations élémentaires telles que READ, WRITE, OPEN, CLOSE, MOVE,etc.

- l'étape texte
  La structure des programmes comportant les opérations allouées est traduite en un pseudo-langage. Les conditions d'itération et de sélection figurant dans la structure sont décrites au cours de cette étape.

- l'étape d'implémentation
  A l'issue de cette étape, le programme est fin prêt pour la mise en production. On établit l'éventuelle connexion avec un autre programme." [ibidem, pp.2-3]

2.3.6. LES TESTS

Les programmeurs sont responsables de la correction de leurs programmmes et du déroulement des tests d'intégration.

Les utilisateurs procèdent aux tests du produit en vue de découvrir si ce dernier répond bien à leurs desiderata. Les spécifications initiales et les structures de données jouent un rôle très important à ce niveau.

En effet, les utilisateurs doivent les étudier pour :
- définir les procédures de test,
- créer l'environnement nécessaire à l'exécution de tests significatifs et, en particulier, générer les données de test,
- prévoir les résultats que devra donner chacun des tests envisagés,
- exécuter les tests. [MAI (b), p.4]

Rappelons qu'à ce niveau, intervient l'ensemble des caractéristiques fonctionnelles définies tout au long de l'analyse.

2.3.7. LE PASSAGE EN PRODUCTION

Lors du développement,le programme se trouve dans une librairie de développement; la passage en production consiste à transférer le programme dans une librairie de production avec un minimum de documentation pour permettre à l'opérateur de lancer le nouveau système (noms des fichiers, leurs caractéristiques, les JCL : Job Control Language,...)

2.4. SYNTHESE

Comme nous venons de le voir, la méthodologie proposée a la CGER est une méthodologie basée avant tout sur les données et ce, d'une part, pour leur plus grande stabilité et, d'autre part, pour des raisons historiques que nous ne reprendrons pas ici. Si la filière des données est précise, la filière des traitements, quant à elle est plus ambiguë. En effet, à ce niveau, il existe un "trou" méthodologique entre l'analyse fonctionnelle et la programmation. Cependant, selon les responsables des méthodes, il semble que la découpe fonctionnelle pourrait être une étape préalable à l'application de la méthode JSP (Jackson Structured Programming) pour la suite du développement.
Par ailleurs, quelques-unes des phases sont sujettes à modification : il s'agit de la définition de projet qui devrait être moins fastidieuse à l'avenir et de la démarche à adopter pour les quantifications de l'analyse fonctionnelle. A ce sujet, nous faisons référence à PALOMA qui sera décrit dans les points de passage (cfr. infra).

3. DESCRIPTION DE LA METHODOLOGIE ETUDIEE A L'INSTITUT D'INFORMATIQUE DE NAMUR (F. Bodart et Y. Pigneur)

La méthodologie proposée par F. Bodart et Y. Pigneur se limite à l'analyse fonctionnelle.
Son but est de décrire les informations, les traitements et les ressources qui constituent les applications de gestion de "grande taille" qui sont aussi appelées systèmes d'information (S.I.). [BODART (a), pp 1-3]

Nous détaillerons tout d'abord les différents modèles qui sont utilisés dans cette méthodologie et, ensuite, nous décrirons en quoi elle consiste. Toutes les citations entre guillemets proviennent de [BODART (a)].

3.1. DESCRIPTION SUCCINCTE DES MODELES

Les différents modèles auxquels il sera fait référence dans l'analyse fonctionnelle sont le modèle de structuration des informations, les modèles de structuration des traitements, de la dynamique et de la statique des traitements, le modèle des ressources et le diagramme des flux de données.

3.1.1. LE MODELE DE STRUCTURATION DES INFORMATIONS

Le but d'un modèle de structuration des informations est de décrire et de définir "rigoureusement" les informations mémorisées dans un système d'information et aussi les messages qui véhiculent les informations. [BODART (a), pp 12-50]

Le modèle qui a été retenu en vue de réaliser cet objectif est le modèle ENTITE-ASSOCIATION qui propose de modéliser les informations du réel perçu à partir de trois concepts élémentaires : entité, association et attribut. Ce modèle est ensuite complété par le modèle des messages.

a) Les concepts de base

"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" (cfr. infra).

Cependant, l'intérêt de la modélisation n'est pas d'individualiser les objets mais d'envisager plutôt des classes d'objets que l'on appelle types d'entités (T.E.). Il doit être toujours possible de donner une définition précise de l'ensemble des entités d'un même type.

"Une association est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné. On veut enregistrer de l'information relative à cette correspondance. Une association peut posséder un ou plusieurs attributs" (cfr. infra).

De la même manière que pour entité et type d'entité, on pourra parler d'association et de type d'association (T.A.). Il doit également être possible de donner une définition précise de l'ensemble des associations d'un même type.

Pour chaque couple (T.E.,T.A.) tel que T.E. est relié à T.A., on peut définir un couple (min,max) où min représente le nombre minimum d'associations de T.A. auxquelles toute entité de T.E. doit participer à tout moment et où max représente le nombre maximum d'associations de T.A. auxquelles toute entité de T.E. peut participer. Ce couple (min,max) est appelé connectivité du couple (T.E.,T.A.).

"Un attribut est une caractéristique ou qualité d'une entité ou d'une association. Elle peut prendre une ou plusieurs valeurs ou groupes de valeurs". Un attribut peut être simple ou répétitif, élémentaire ou décomposable, obligatoire ou facultatif.

Ces trois concepts de base peuvent être étendus par la notion de contrainte d'intégrité.

b) Les contraintes d'intégrité

"Une contrainte d'intégrité est une propriété, non représentée par les concepts de base du modèle que doivent satisfaire les informations appartenant à la mémoire du S.I".

Mais en fait, on peut dégager une infinité de contraintes d'intégrité pour tout problème aussi on ne conserve que les plus importantes et les plus prédominantes dans la mesure où "elles jouent un rôle essentiel dans la conception d'un S.I."

- Identifiant

L'identifiant attaché à un type d'entité ou à un type d'association permet de repérer univoquement chaque occurrence de ce type.

En particulier :
- l'identifiant d'un T.E. est composé d'un ou plusieurs attributs qui appartiennent au T.E. à identifier et/ou aux T.E. qui lui sont reliés via des T.A.;
- l'identifiant d'un T.A. est composé des identifiants des T.E. sur lesquels il est défini. De plus, si cela s'avère nécessaire un T.A. peut se voir attribuer un identifiant propre de la même manière que pour un T.E.

- Autres contraintes

Les autres contraintes portant sur les T.E., T.A. et attributs servent à exprimer toutes les propriétés sémantiques non reprises dans le modèle et dont le concepteur du S.I. voudrait qu'il soit tenu compte pour garantir la cohérence et la validité de la base de données.

Parmi ces contraintes d'intégrité, citons les dépendances fonctionnelles qui nous semblent assez particulières pour faire l'objet d'une description plus approfondie.

- Les dépendances fonctionnelles

"Etant donné un T.E. ou un T.A., un attribut B dépend fonctionnellement d'un attribut A, si, à tout moment, à chaque valeur de A correspond au plus une valeur de B".

"Les dépendances fonctionnelles jouent un rôle constructif dans l'élaboration d'un schéma conceptuel des données. Elles sont utilisées notamment pour vérifier si une structure de données n'est pas désagrégable ou décomposable en structures plus simples qui exprimeront parfois de façon plus précise les faits et les concepts d'une organisation".

c) La représentation graphique

Le symbolisme du modèle Entité-Association est le suivant :

- les types d'entités sont représentés par des rectangles qui contiennent leur nom et leurs attributs. L(es) attribut(s) identifiant(s) est (sont) souligné(s);

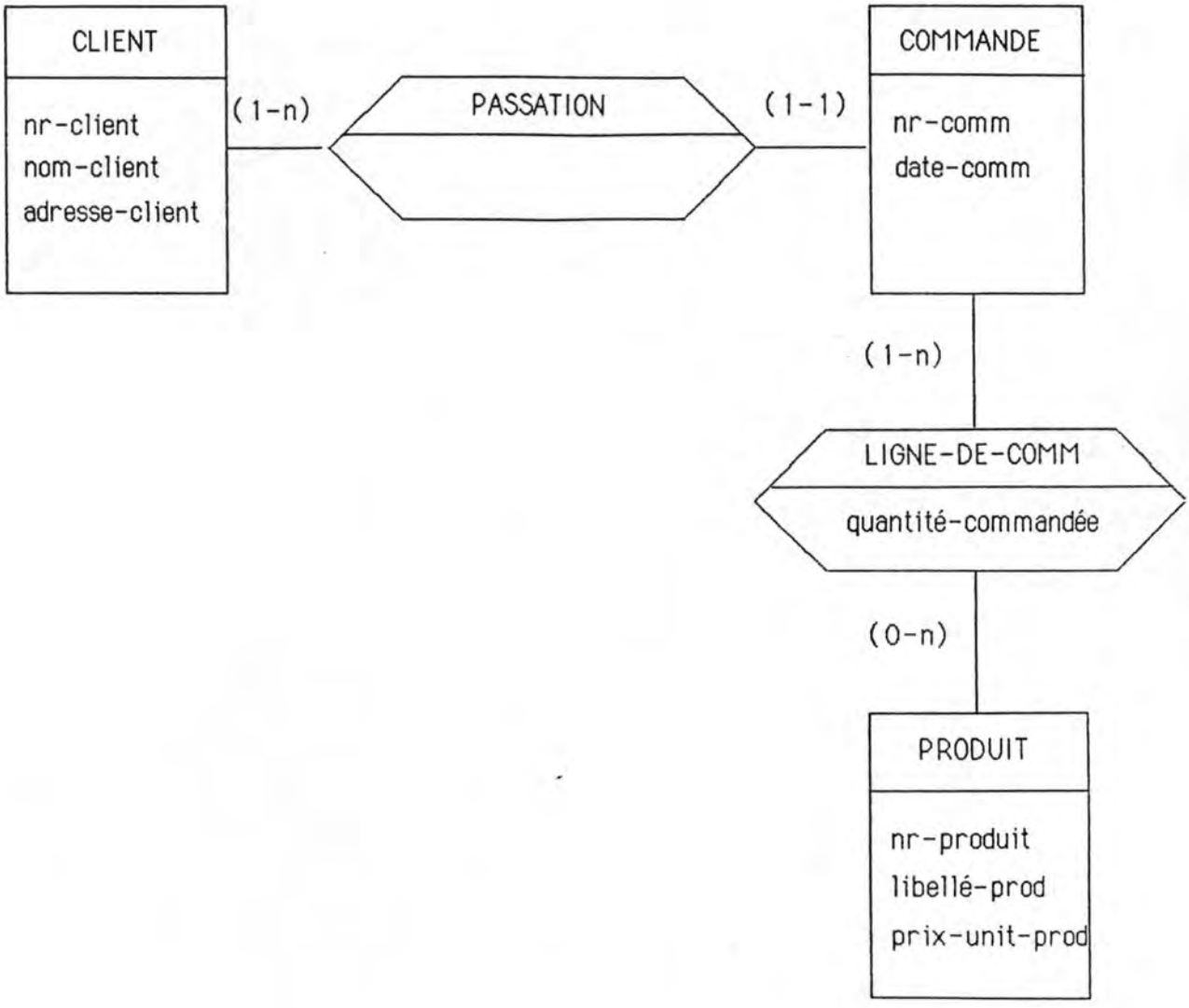
- les types d'associations sont représentés par des hexagones qui contiennent leur nom et leurs attributs. L(es) attribut(s) identifiant(s) est (sont) souligné(s);

- les types d'entités sur lesquels sont définis les types d'associations sont reliés à ceux-ci par un trait accompagné du couple (min,max) précisant la connectivité.

Considérons l'énoncé qui suit :

Un client est toute personne qui a passé au moins une commande à la firme X. Chaque client est identifié par un numéro et est aussi décrit par un nom et une adresse. Chaque commande est identifiée par un numéro et est composée d'une date et d'un ensemble de lignes décrivant chacune le produit et la quantité commandée de ce produit. Chaque produit est identifié par un numéro et comporte aussi par un libellé et un prix unitaire.

Ces informations peuvent être représentées de la façon suivante :



Remarque :

1. "La représentation graphique"..."est destinée à faciliter la communication. Elle est incomplète et à la limite ne signifie rien si elle n'est pas complétée par la définition des types d'entités, types d'associations et attributs". Si nous voulons rester fidèles au modèle, les contraintes d'intégrité doivent aussi être mentionnées, ce qui entraîne la remarque suivante.

2. Si la définition des types d'entités et des types d'associations est complète, il y a obligatoirement des redondances entre les définitions et les contraintes d'intégrité. Reprenons l'exemple ci-dessus et supposons qu'"une occurrence de COMMANDE ne peut exister que si elle participe à au moins une ligne de commande telle que la quantité commandée soit supérieure à 0". Cette propriété est vue par F. Bodart et Y. Pigneur [BODART (a), p 39] comme une contrainte d'intégrité alors que nous préférerions la considérer comme faisant partie de la définition du T.E. COMMANDE.

3. La représentation du réel perçu est un choix partiellement arbitraire. En effet, la représentation du réel perçu peut être différente d'un concepteur à l'autre selon leur propre vision des choses. De plus, un même concepteur peut imaginer plusieurs schéma pour un même problème.

d) Le modèle des messages

Le modèle de structuration des informations doit aussi contenir la description des messages qui circulent dans un S.I. Ceux-ci devront être définis en terme des informations qu'ils véhiculent.

3.1.2. LE MODELE DE STRUCTURATION DES TRAITEMENTS

Il correspond à une démarche par raffinements successifs. Il consiste en effet à décomposer un traitement en traitements plus simples dans le but de former une arborescence à 4 niveaux. Ainsi, un traitement de niveau inférieur "fait partie" d'un traitement de niveau supérieur. [BODART (a), pp 50-59]

La nomenclature définie sur les traitements des différents niveaux est la suivante :

Au niveau 1, "le projet est la partie du système d'information qui fait l'objet d'une analyse".

Au niveau 2, "l'application est un traitement quasi-autonome par rapport aux autres applications d'un projet. Elle représente la dimension minimale d'un projet informatique".

Au niveau 3, "la phase est un traitement (manuel ou automatisable) possédant une unité spatio/temporelle d'exécution", c'est-à-dire réalisé sans interruption, sans changement de lieu ni de ressources. La phase est un repère central car elle constitue notamment "un lieu d'identification de structures homogènes de données et de traitements".

Au niveau 4, "la fonction correspond au niveau élémentaire de la nomenclature des traitements ; elle est associée à un objectif et à un comportement organisationnel considérés comme élémentaires par l'organisation".

Ces définitions permettent de dégager un certain nombre de critères d'identification qui serviront de guide à la décomposition.

3.1.3. LE MODELE DE LA DYNAMIQUE DES TRAITEMENTS

Il fournit des concepts et des mécanismes permettant de représenter les conditions de déclenchement, d'exécution et d'enchaînement des traitements. [BODART (a), pp.59-81]

Interviennent ici les notions de processus et d'événement:

"Un processus est l'exécution d'une procédure de traitement de l'information dont la progression peut être représentée, à des points dans le temps, par son état". Les trois états possibles sont : l'état déclenché (c'est-à-dire créé mais en attente d'exécution), l'état actif et l'état terminé. Un processus sera noté Pi.

"Un événement correspond à un changement d'état du système d'information localisé dans le temps et dans l'espace". Les trois changements d'état possibles sont : le déclenchement, l'activation et la terminaison.

Ces deux concepts seront combinés à l'aide de structures d'enchaînement des traitements et de règles y associées. Les structures d'enchaînement des traitements sont :

- l'enchaînement séquentiel : la terminaison de P1 provoque le déclenchement de P2;
- l'enchaînement éclaté : la terminaison de P1 provoque le déclenchement simultané de P2...Pn;

- l'enchaînement multiple : la terminaison de P1 provoque le déclenchement simultané de n processus de type P2;

- l'enchaînement conditionnel : la terminaison de P1 déclenche le processus P2 si une certaine condition est vraie et P3 sinon;

- l'enchaînement convergent : le déclenchement d'un processus Pi est provoqué par la terminaison d'un des processus P1 ou P2 ou ... Pn;

- l'enchaînement synchronisé : le déclenchement d'un processus Pi est provoqué par la terminaison des processus P1 et P2 et ... Pn.

Certains de ces enchaînements sont assujettis à des mécanismes de synchronisation, de condition et de déclenchement multiple qui devront faire l'objet d'une spécification complète.

Remarque : le modèle de la dynamique des traitements suppose que l'on ait d'abord réalisé le modèle de la structuration des traitements.

3.1.4. LE MODELE DE LA STATIQUE DES TRAITEMENTS

"Après avoir structuré les traitements, analysé leur dynamique et défini le schéma conceptuel des informations, on procède à la spécification de la statique des traitements". [BODART (a), pp 82-85]

"Ce modèle permet de spécifier pour un traitement de niveau donné :
- les objectifs à réaliser,
- les performances souhaitées,
- les informations en entrée et en sortie,
- les actions primitives qui caractérisent les relations entre ces informations et le traitement".

On complète la spécification de chaque traitement en décrivant les règles d'obtention des informations de sortie à partir des informations d'entrée (règles de traitement).

En cas de traitements interactifs, on complète la description des règles de traitement par la description du dialogue utilisateur-terminal.

La forme de description la plus appropriée pour ce modèle est la langue naturelle.

3.1.5. LE MODELE DES RESSOURCES

Il sert à caractériser les ressources réutilisables ou consommables nécessaires à l'exécution des traitements, ceci permettra "d'évaluer le caractère réalisable d'une solution conceptuelle". [BODART (a), pp 85-91] (cfr. chapitre 2, IDA).

Dans ce but, toute ressource doit être décrite par la spécification des traitements qui la requièrent ou la partagent si c'est une ressource réutilisable, ou des traitements qui la consomment si c'est une ressource consommable.

3.1.6. LE DIAGRAMME DES FLUX DE DONNEES

Le diagramme des flux est un outil de représentation du fonctionnement du S.I. et de communication avec l'utilisateur. [BODART (a), pp 147-150]

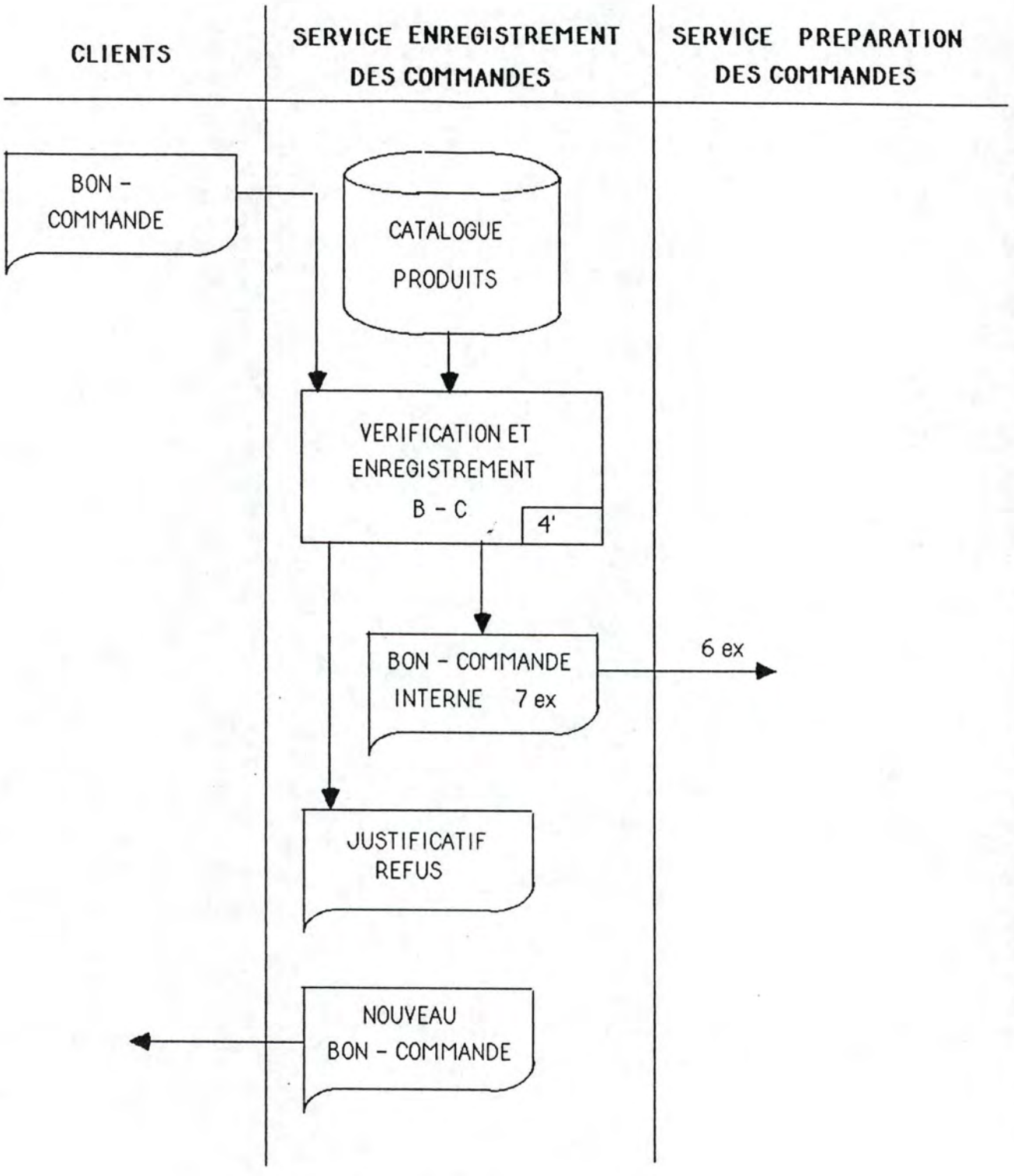
Un diagramme des flux :

- "décrit le cheminement des messages, leurs origines et leurs destinations";
- "fournit une localisation spatiale et temporelle des "points" de naissance, de transformation et de disparition des informations";
- "indique, pour chaque traitement (de type phase), les messages (avec éventuellement le nombre d'exemplaires créés) et les informations mémorisées en entrée et en sortie";
- "indique la durée unitaire d'un traitement (si possible) ainsi que le délai qui sépare deux traitements distincts effectués en séquence".

Considérons l'énoncé suivant [BODART (a), p 149] :

Un client émet un bon de commande qui est traité dans le service d'enregistrement des commandes qui procède à la vérification et à l'enregistrement du bon de commande. Pour réaliser ce traitement, on dispose en entrée du catalogue des produits. Une commande refusée est renvoyée au client avec un justificatif de refus et un nouveau bon de commande. Une commande acceptée donne lieu à la création en 7 exemplaires d'un bon de commande interne. La durée d'une opération de vérification et d'enregistrement du bon de commande est estimée à 4 minutes. Six exemplaires du bon de commande interne sont adressés au service de préparation des commandes qui procède à la préparation de la livraison (prélèvement des produits et préparation des documents).

Ce qui donne comme représentation :



Pour construire le diagramme des flux, il faut au préalable élaborer les modèles de structuration des traitements, de la dynamique et de la statique des traitements ainsi que le modèle des ressources. Chacun de ces modèles apporte des éléments complémentaires à la construction du diagramme des flux :

- le modèle de structuration des traitements fournit les traitements définis comme "phases" qui sont représentées par des rectangles;

- le modèle de la dynamique des traitements fournit le cheminement des messages via les postes de travail et les traitements. Un des axes du diagramme est en effet associé au cheminement des messages et leurs flux sont représentés par des flèches sur lesquelles le nombre d'exemplaires transmis est éventuellement indiqués;

- le modèle de la statique des traitements fournit les entrées et les sorties d'un traitement ainsi que l'origine et la destination des messages; les informations mémorisées sont représentées par un cylindre;

- le modèle des ressources fournit les spécifications des postes de travail où sont exécutés les traitements ainsi que celles des durées et des ressources qu'ils utilisent; chaque colonne du diagramme représente un poste de travail tandis que c'est au niveau des traitements que les durées peuvent être mentionnées.

Remarques :

1. les informations en E/S sont désignées globalement sous forme de collections d'informations (exemple : le catalogue des produits) ou sous forme de fichier.

2. "dans un but de simplicité, on se limitera à décrire des enchaînements séquentiels. Cependant, dans certains cas, il peut être opportun d'introduire des structures de contrôle conditionnelles, à déclenchements multiples, synchronisées".

3.2. DESCRIPTION DE LA METHODOLOGIE

La méthodologie proposée par F. Bodart et Y. Pigneur se limite, comme nous l'avons déjà souligné, à l'analyse fonctionnelle. Cette analyse se compose de deux parties : l'étude d'opportunité et l'analyse conceptuelle. Nous allons à présent les étudier.

3.2.1. L'ETUDE D'OPPORTUNITE

Son objectif est de préparer un avant-projet de solution à partir des besoins exprimés par l'organisation et ce, dès le moment où une décision d'analyse a été approuvée. [BODART (a), pp 138-154]

L'étude d'opportunité se compose de plusieurs phases : l'identification du projet, la définition du projet-cadre et l'étude critique du S.I. existant, l'élaboration de solutions, le choix d'une solution et la mise à jour du projet-cadre. (cfr. fig. 1, page 37)

a) Identification du projet

Le projet sera identifié entre autres par les causes profondes d'insatisfaction du système existant s'il y en a; dans ce cas, on spécifiera également la localisation de ces déficiences. De toute façon, on s'efforcera de délimiter la frontière du projet en identifiant les messages en provenance et à destination de l'environnement de ce projet.

b) Définition du projet-cadre

Il contient la spécification des besoins exprimés sous forme d'objectifs organisationnels (comportements opérationnels ou comportements de gestion de l'organisation) et informationnels (qualité de l'information et qualité des traitements de ces informations).

Ensuite, on mentionnera les activités concernées par les modifications apportées au S.I. Ces activités correspondent aux phases qui seront définies dans l'étude critique du S.I. De ce fait, on pourrait supposer que cette phase est postérieure à la phase de l'étude critique du S.I. existant. On verra ce qu'il en est réellement au point suivant.

Finalement, on fixera les critères ainsi que les contraintes en vue, d'une part, d'évaluer les différentes solutions qui seront proposées et, d'autre part, de constituer un point de référence pour le contrôle de la mise en oeuvre et de l'exploitation du projet.

c) Etude critique du S.I. existant

Pour procéder à l'étude critique du S.I. existant, il serait utile de disposer d'un outil simple et apte à représenter synthétiquement le fonctionnement de ce S.I. Aussi, F. Bodart et Y. Pigneur proposent-ils le diagramme de flux de données.

La construction du diagramme sera la première chose à faire et l'analyse critique de l'existant consistera alors à dégager les disfonctionnements du S.I. sur base de ce diagramme. Cette critique "doit tenir compte des objectifs et des besoins énoncés dans le projet-cadre et, réciproquement, la définition des objectifs à atteindre ne peut être indépendante des caractéristiques du S.I. existant". Ceci nous amène au fait qu'il n'y a pas de relation d'ordre entre la définition du projet-cadre et l'étude critique de l'existant mais bien de nombreuses interactions. Celles-ci sont traduites sur le schéma d'ordonnancement des phases de l'analyse d'opportunité (p. 37) par leur réalisation en parallèle.

d) Elaboration de solutions

"Elaborer une solution consiste à redéfinir les activités et les règles de traitement de l'information, à envisager des modifications technologiques, à redéfinir les fonctions et les rôles qui seront assumés par les personnes et à repenser les structures d'organisation."

Donc, l'élaboration de solutions reposera sur des stratégies jouant sur les dimensions suivantes : la définition fonctionnelle des traitements, la technologie, les personnes et les structures d'organisation.

Chaque solution sera exprimée par un diagramme des flux complété par les objectifs, les performances et les ressources associés à chaque phase mais aussi par un relevé "des principales modifications des fonctions de l'organisation et des rôles assumés par les personnes".

e) Choix d'une solution

On évaluera ici les différentes solutions proposées sur base des critères définis dans le projet-cadre. L'évaluation pourra être facilitée par l'utilisation de modèles tels que le calcul de rentabilité, le modèle multicritères ou l'analyse coût-efficacité. De toute façon, le choix reste du domaine du décideur.

f) Mise à jour du projet-cadre

Cette mise à jour consiste en un "raffinement" du projet-cadre sur base de la solution choisie. Dans ce but, on mentionnera les objectifs poursuivis par l'organisation, les efficacités attendues de la solution retenue, l'exposé de la solution adoptée, les quantifications pour l'élaboration du cahier des charges, le planning et les points de contrôle dans la réalisation du projet.

Remarque : la définition d'un point de contrôle n'est mentionnée nulle part, la seule information dont nous disposons est qu'un point de contrôle est caractérisé "notamment" par une documentation appropriée.

L'ordonnancement de ces différentes étapes est représenté ci-dessous (fig. 1) et nous pouvons remarquer qu'à l'issue de l'analyse d'opportunité, on pourra disposer du projet-cadre mis à jour.

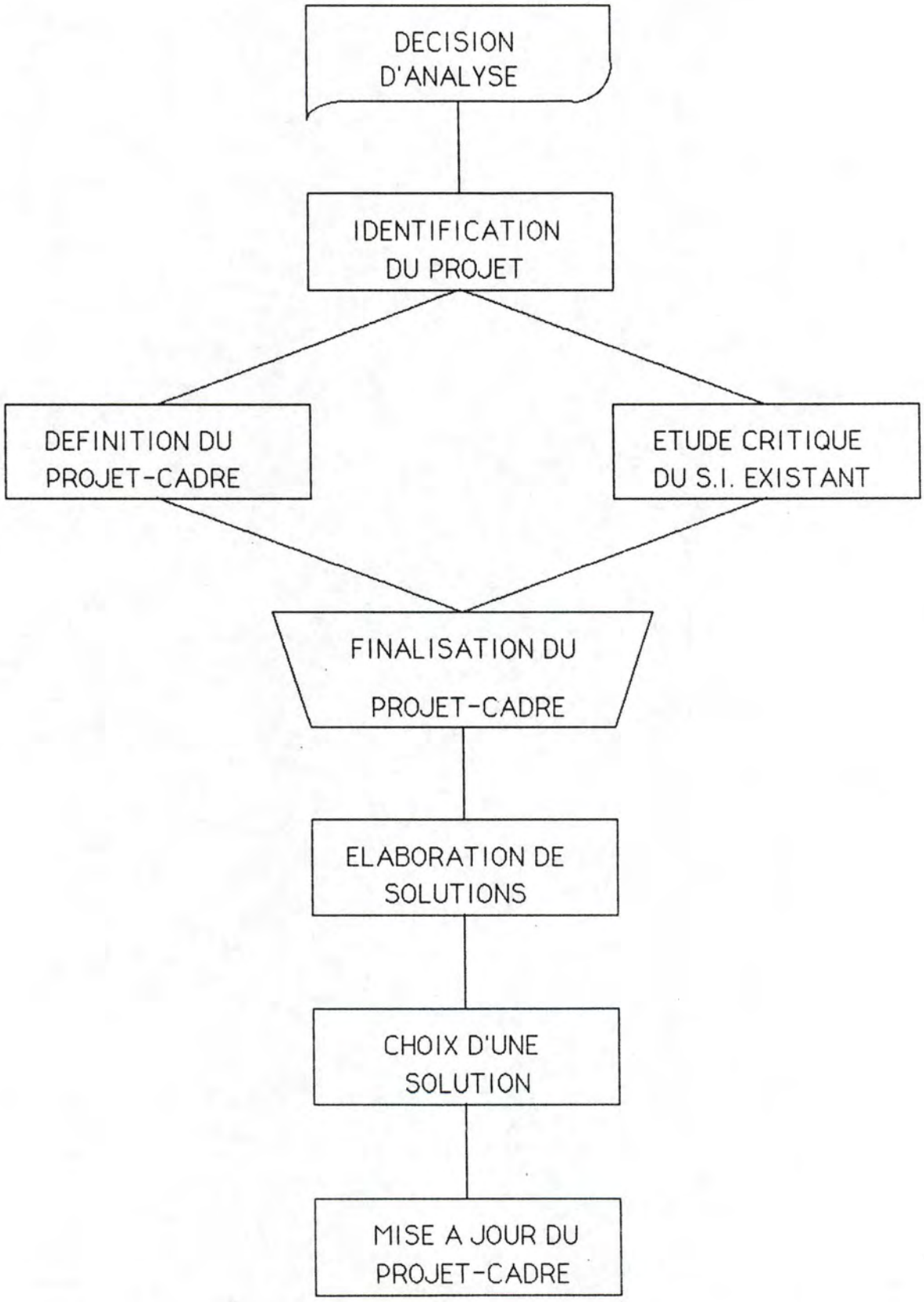


fig. 1 : ordonnancement des phases de l'analyse d'opportunité

3.2.2. L'ANALYSE CONCEPTUELLE

Elle élabore sur base du projet-cadre une solution (conceptuelle) détaillée mais indépendante de tout moyen de réalisation. [BODART (a), pp 155-181]

L'analyse conceptuelle distingue deux étapes majeures :

- l'élaboration et la validation des sous-schémas conceptuels pour chaque "phase" du "projet" comprenant des opérations de traitement de l'information partiellement ou totalement automatisables;
- l'élaboration et la validation du schéma conceptuel global par consolidation des sous-schémas relatifs aux différentes "phases" du "projet".

Ces deux étapes se complètent par l'établissement de quantifications.

a) Elaboration du sous-schéma conceptuel d'une "phase"

Elle comprend deux parties : l'une relative aux "informations" et l'autre relative aux "traitements".

- Elaboration d'un sous-schéma conceptuel des informations

  Pour chaque phase, on construit un schéma Entité/Association brut à partir des différentes informations obtenues principalement lors de l'analyse d'opportunité. Pour chaque sous-schéma brut ainsi obtenu, on procède à "une élimination ou au contrôle de la redondance". Il faut aussi définir tous les éléments intervenant dans le schéma Entité/Association à savoir les entités, les associations et leurs attributs éventuels mais aussi les messages identifiés lors de la construction du schéma. On obtient de la sorte un modèle de structuration des informations pour chaque phase.

- Elaboration d'un sous-schéma conceptuel des traitements

  Pour chaque phase, on procède tout d'abord à sa décomposition, ensuite, on élabore le schéma de la dynamique des fonctions ainsi obtenues en vérifiant qu'il s'insère bien dans le schéma de la dynamique des phases. Et finalement, on élabore un modèle de la statique pour chaque fonction de la phase et pour la phase elle-même.

- Validation du sous-schéma conceptuel d'une phase

  On effectuera ici les contrôles de cohérence et de complétude sur base de la spécification complète de toutes les fonctions de la phase.

b) Elaboration du schéma complet par consolidation des sous-schémas

On retrouve ici aussi la dichotomie information-traitement.

- Consolidation des sous-schémas conceptuels des informations

  "La consolidation des sous-schémas conceptuels des informations est un processus itératif qui, partant d'un schéma conceptuel des informations initial et d'un sous-schéma conceptuel, donne pour résultat un schéma conceptuel consolidé".

  La consolidation consiste également à appliquer des règles relatives à la vérification de la cohérence du schéma résultant. Citons, par exemple, les règles de consolidation des synonymes, les règles de résolution des conflits,...

- Consolidation des sous-schémas conceptuels des traitements

  Les "phases" étant obtenues par partitionnement du "projet", la consolidation des sous-schémas des traitements consiste en une simple juxtaposition des différents sous-schémas conceptuels des traitements.

c) Quantifications

Des quantifications sont établies pour les données, les traitements, les messages et les opérations sur la base de données.

Si on a déjà évalué la quantité de ressources que l'organisation devra affecter à la mise en oeuvre de la solution conceptuelle, cette estimation pourra être complétée par les quantifications sus-mentionnées.

4. COMPARAISON DE LA METHODOLOGIE PROPOSEE PAR LA CGER AVEC LA METHODOLOGIE DECRITE PAR F. BODART ET Y. PIGNEUR

4.1. PORTEE DE LA COMPARAISON

Cette comparaison devra être menée sur un terrain restreint déterminé par la portée de chacune des méthodologies.

Tout d'abord, la méthodologie proposée par F. Bodart et Y. pigneur n'a été décrite que pour "l'analyse fonctionnelle" (qui couvre l'étude d'opportunité et l'analyse conceptuelle), de ce fait, les seules étapes définies par la CGER dont nous tiendrons compte dans notre comparaison sont : la demande de développement, la définition de projet, l'analyse conceptuelle et une partie de l'analyse fonctionnelle.

F. Bodart et Y. Pigneur limitent leur méthodologie à la définition et au contrôle du contenu d'un S.I., ils restent volontairement à la frontière des problèmes dus au "contrôle du processus de développement du S.I. en ce compris le mode de participation des utilisateurs à ce processus" [BODART (a), p 137]. Ainsi, F. Bodart ne donne aucune indication sur la fonction qu'assument dans l'organisation, les personnes qui mènent l'analyse fonctionnelle. Nous avons considéré, par conséquent, que cette analyse était menée, du début à la fin par un même groupe de personnes entièrement responsables de la bonne conduite du projet. De ce fait, les points de décision (autres que la "décision d'analyse" et "le choix d'une solution") doivent être considérés comme implicitement inclus au sein de chaque phase. Dans notre comparaison, il ne nous sera donc pas possible d'envisager le rôle des personnes devant participer au développement du projet.

De part et d'autre, il existe des "bouclages" entre les phases. Un "bouclage" en cours de développement est un retour vers une phase antérieure qui doit être remise en question. La terminaison de chaque phase essentielle constitue un "point de contrôle".

Selon F. Bodart, un point de contrôle est "caractérisé notamment par l'établissement d'une documentation appropriée" [BODART (a), p 139]. Les points de contrôle essentiels sont : l'identification du projet, la finalisation du projet-cadre, la mise à jour du projet-cadre, la consolidation des sous-schémas conceptuels des informations et la consolidation des sous-schémas conceptuels des traitements. Rien n'est dit au sujet des responsables d'un contrôle éventuel, ce qui est logique puisque l'on a considéré que les responsables du développement travaillent ensemble pour chacune des phases.

A la CGER, les points de contrôle sont appelés "points de passage". Leur fonction est d'exiger une trace écrite prouvant que telle phase de la méthodologie a bien été élaborée par le groupe de travail. Actuellement, les points de passage sont : la demande de développement, la définition de projet, le schéma conceptuel des données et celui des traitements.

4.2. DEMARCHE ADOPTEE POUR LA COMPARAISON

Notre comparaison consistera à situer et à justifier, si possible, les points de divergence. Pour atteindre cet objectif, nous avons tenté d'établir un parallélisme entre les phases qui, de part et d'autre, présentent des analogies sur base de leur finalité.

Ainsi, notre comparaison se déroulera en 4 parties : nous envisagerons, tout d'abord, la décision d'analyse et l'identification de projet (F. Bodart) avec la demande de développement (CGER); ensuite, nous examinerons la définition du projet-cadre et l'étude critique de l'existant (F.Bodart) en parallèle avec l'étude de l'existant et la définition des objectifs et contraintes du nouveau système (CGER); puis nous envisagerons la description des solutions et le choix d'une solution de part et d'autre; finalement, nous comparerons les deux analyses conceptuelles où nous ferons intervenir des éléments de l'analyse fonctionnelle de la CGER.

Schéma de la démarche :

F. Bodart et Y. Pigneur	CGER
- décision d'analyse - étude d'opportunité : . identification de projet (cfr. 4.3.1.)	- demande de développement
. définition projet-cadre . étude de l'existant (cfr. 4.3.2.)	- définition de projet : . étude de l'existant . objectifs, contraintes
. solutions possibles . choix d'une solution . MAJ du projet-cadre (cfr. 4.3.3.)	. solutions possibles . choix d'une solution
- analyse conceptuelle (cfr. 4.3.4.)	- analyse conceptuelle - analyse fonctionnelle

4.3. COMPARAISON

4.3.1. DECISION D'ANALYSE ET IDENTIFICATION DE PROJET (F. Bodart) VS DEMANDE DE DEVELOPPEMENT (CGER)

Les informations reprises dans l'identification de projet sont également incluses dans la demande de développement. Parmi ces informations, figure chez François Bodart la définition de la frontière du projet. Cette frontière est constituée des activités internes au système ayant des liens avec les flux d'informations en provenance et à destination de l'environnement. Le projet est donc l'ensemble des activités situées en deçà de cette frontière. Dans la demande de développement, cette frontière est moins explicite, elle est déductible de la description du contexte actuel.

Outre ces informations, la demande de développement mentionne aussi l'ampleur du projet (volumes d'informations à traiter), les avantages attendus (quantifiables ou non) ainsi que les contraintes de réalisation (planning et budget). Ce type d'indication se retrouve dans le projet-cadre chez François Bodart, nous y reviendrons dans un point ultérieur.

La différence quant au contenu de ces deux documents peut être justifiée par les objectifs qui leur sont assignés. En effet, la demande de développement doit contenir toutes les informations nécessaires (dont les objectifs et contraintes) pour permettre aux responsables d'approuver ou non le développement du nouveau projet. Pour François Bodart, la décision d'analyse est considérée comme un message qui déclenche la phase d'identification de projet. L'identification de projet, quant à elle, sert à délimiter le champ d'action du nouveau système ainsi qu'à souligner les raisons sous-jacentes à la décision de développer ce système (causes d'insatisfaction). Cette décision est donc prise avant la phase d'identification de projet.

4.3.2. DEFINITION DU PROJET-CADRE ET ETUDE CRITIQUE DE L'EXISTANT (F. Bodart) VS ETUDE DE L'EXISTANT ET DEFINITION DES OBJECTIFS ET CONTRAINTES (CGER)

La définition du projet-cadre et l'étude critique de l'existant s'effectuent en parallèle, néanmoins, le processus peut être décomposé en 3 étapes que l'on mettra en relation avec leur correspondant à la CGER.

Premièrement, F. Bodart propose de représenter le système existant par un formalisme simple et communicable : le diagramme des flux de données. Rappelons que ce diagramme est constitué à partir des informations obtenues après application de différents modèles de description du S.I. existant. Ces modèles sont ceux de la structuration, de la statique et de la dynamique des traitements (appliqués jusqu'au niveau de la "phase") et celui des ressources. Le modèle conceptuel des données n'est pas indispensable ici puisque les données sont désignées globalement selon des collections logiques ou physiques.

A cette dernière étape de la démarche correspond, à la CGER, l'étude de l'existant où le résultat est également un diagramme de flux de données documenté (et ce, via le recensement des activités et tâches ainsi que des documents et fichiers). Notons que l'aspect des ressources n'est pas envisagé.
Ici, aucune méthode n'est proposée pour la réalisation des recensements ni pour la construction du diagramme. Ceci peut s'expliquer par le fait qu'à la CGER les développements de projets sont, dans la plupart des cas, soit des modifications, soit des extensions d'un système. Si le système initial a été développé selon la même méthodologie, alors la représentation de l'existant peut être facilitée par les documents rédigés lors du développement de ce dernier système.

Cependant, établir et appliquer des règles de construction d'un diagramme de flux de données est loin d'être une sinécure. En effet, si François Bodart affirme que "le modèle de la dynamique des traitements constitue la base d'un diagramme des flux" puisqu'il "il permet de décrire le cheminement des messages via les postes de travail et les traitements" [BODART (a), p 150], l'expérience nous montre dans quelle mesure surgissent les problèmes de communication avec l'utilisateur quand il faut supprimer une partie des structures de contrôle et ce, en vue d'obtenir un schéma clair.

La seconde étape de la démarche de François Bodart consiste à définir les objectifs de l'organisation et les objectifs informationnels ainsi qu'à spécifier les activités concernées par ces objectifs. Ceci se fait sur base du diagramme de flux de données du S.I. existant. A ce stade, on mentionnera également une série de critères d'efficacité qui seront évalués par la suite et traduits en contraintes.

La troisième étape consiste à critiquer le S.I. existant. Il sera procédé à une critique fonctionnelle où l'on évalue "le S.I. existant en termes des efficacités informationnelles, organisationnelles et économiques" [BODART (a), p 150] (définies dans la

seconde étape). Le système sera également critiqué d'un point de vue structurel, en vue de déceler les lacunes, les anomalies et les disfonctionnements.

La réalisation de ces trois étapes conduit à la finalisation du projet-cadre chez François Bodart.

A la CGER, la critique du S.I. existant est effectuée avant la détermination des objectifs et des contraintes. De plus, cette critique est uniquement structurelle. Cette inversion par rapport aux étapes 2 et 3 chez François Bodart est peut être due au fait qu'à la CGER certains objectifs et contraintes sont définis dans la demande de développement, ces derniers permettant de fonder la critique. Néanmoins, étant donné que ces objectifs et contraintes sont plus généraux et qu'ils ne reposent pas sur une analyse structurelle et organisationnelle du système, il est souhaitable de les redéfinir de manière plus précise afin de permettre une critique plus pertinente.

4.3.3. ELABORATION DE SOLUTIONS ET CHOIX D'UNE SOLUTION

De part et d'autre, il est procédé à l'élaboration des solutions possibles. La description utilisée à la CGER est plus détaillée. En effet, elle comporte en plus une analyse des coûts de chaque solution ainsi qu'un planning de réalisation.

Par ailleurs, François Bodart insiste sur les dimensions à prendre en considération. "Il faut envisager non seulement les modifications technologiques mais aussi les structures d'organisation et les fonctions et les rôles qui seront assignés aux personnes". [BODART (a), p 151].

A la CGER, les décideurs disposent directement de toutes les informations pour effectuer un choix. Aussi, la solution choisie sera parfaitement définie pour les étapes ultérieures du développement.

Pour François Bodart, ce n'est qu'au moment du choix que l'on évalue les solutions par rapport aux critères d'efficacité énoncés dans la définition du projet-cadre. Les calculs de rentabilité, les analyses coût-efficacité ou l'approche multicritères guideront les décideurs dans leur choix.

Le projet-cadre précédemment élaboré sera mis à jour en fonction des efficacités attendues de la solution choisie, un planning et un cahier des charges seront mis au point.

Ainsi, ce projet-cadre mis à jour pour une solution chez François Bodart correspond à la description de la solution choisie à la CGER, description qui aura été effectuée pour chacune des solutions proposées.

4.3.4. ANALYSE CONCEPTUELLE (F. Bodart) VS ANALYSE CONCEPTUELLE ET ANALYSE FONCTIONNELLE (CGER)

Nous comparerons, tout d'abord, les deux démarches d'analyse conceptuelle; ensuite, nous dégagerons les éléments de l'analyse conceptuelle chez F. Bodart qui interviennent dans l'analyse fonctionnelle à la CGER.

a) L'analyse conceptuelle

Au niveau de l'analyse des données, on peut considérer que les résultats obtenus sont équivalents. En effet, de part et d'autre, on aboutit à un modèle Entité-Association des données. Cependant, la démarche proposée à la CGER pour construire ce modèle est une démarche bottom-up alors que celle-ci serait plutôt top-down chez F. Bodart. De plus, F. Bodart propose d'élaborer un sous-schéma par phase et de consolider ensuite tous les sous-schémas obtenus en un schéma global des données. Cette méthodologie propose également un modèle de spécification des messages, ce que l'on ne retrouve pas à la CGER.

Au niveau de l'analyse des traitements, il existe deux nomenclatures différentes : le système, l'activité et la tâche à la CGER et le projet, l'application, la phase et la fonction chez F. Bodart. Nous n'avons pas tenté de comparer ces deux nomenclatures car les conclusions que nous pourrions tirer d'une telle comparaison risquent de ne pas reposer sur des bases solides vu la divergences des interprétations que l'on peut donner aux définitions et aux critères d'identification de chacun des concepts. Par ailleurs, il faut au préalable s'interroger sur la pertinence de cette comparaison étant donné que les principes sous-jacents aux différentes démarches sont les mêmes des deux côtés. C'est ce que nous tenterons de développer ci-dessous.

Le but du modèle de structuration des traitements est d'arriver, via une décomposition des traitements par raffinements successifs, à une arborescence de ces traitements.

Le but du modèle de la statique des traitements chez F. Bodart et de l'élaboration des schémas HIPO à la CGER est de décrire le plus complètement possible tous les traitements d'un même niveau (à savoir respectivement celui de la phase et celui de la tâche).

Soulignons les différences entre ces deux types de description des traitements. Le schéma HIPO reprend le même type d'informations que le modèle statique des traitements mais, en outre, il mentionne l'origine et la destination des données ainsi que des informations de type dynamique : les évènements initiateurs et induits. Ceci s'explique par le fait que la méthodologie de la CGER ne dispose pas de modèle de la dynamique au sens du modèle de la dynamique décrit par F. Bodart.

Une fois la description de ces traitements (phase ou tâche) établie, il est procédé de part et d'autre à la consolidation. Cela consiste à regrouper les différentes descriptions et à effectuer tous les contrôles de cohérence et de complétude requis (et ce, sur base également du modèle de structuration des informations).

Les modèles utilisés au niveau de la dynamique des traitements sont : le modèle de la dynamique pour la phase et l'application chez F. Bodart et la notion de processus à la CGER. Ces deux concepts sont différents, il ne faut donc pas en rechercher les points communs.

b) L'analyse fonctionnelle

La méthodologie décrite par F. Bodart s'arrête à la phase d'analyse conceptuelle, ce qui constitue une contrainte pour notre comparaison. Néanmoins, l'analyse conceptuelle (chez F. Bodart) inclut aussi une série de quantifications. Or la quantification est une opération qui a plutôt trait au fonctionnement du système. C'est pourquoi, on retrouve cette étape dans l'analyse fonctionnelle à la CGER.
Les quantifications chez F. Bodart portent sur le volume des données, sur la fréquence des traitements, sur le nombre d'opérations effectuées sur la base de données, sur les ressources ainsi que sur le volume du trafic à écouler sur les canaux de communication (messages).

De la même manière, on retrouve la spécification des dialogues utilisateur-système dans l'analyse fonctionnelle à la CGER et dans le modèle de la statique qui fait partie de l'analyse conceptuelle chez F. Bodart.

5. APPLICATION DE LA METHODOLOGIE A LA CGER

Maintenant que nous avons décrit la méthodologie proposée par la CGER, il nous semble intéressant de constater l'usage qu'il en est fait par les développeurs de projets. Mais avant d'aborder ce sujet, il serait utile de comprendre pourquoi la méthodologie n'est pas suivie de façon stricte par les développeurs.

5.1. LA METHODOLOGIE N'EST PAS APPLIQUEE A LA LETTRE

Il est rare que des projets soient développés par application à la lettre de cette méthodologie. Des contacts que nous avons eus avec quelques développeurs et responsables des méthodes, nous pouvons déduire certains arguments susceptibles de justifier ce comportement :

a) De par la diversification de leur expérience respective, les développeurs sont rarement enclins à utiliser une méthodologie qui leur est imposée si celle-ci ne correspond pas à la démarche qu'ils ont l'habitude d'utiliser.

b) L'application de règles précises imposées par une méthodologie est souvent considérée par les développeurs comme un obstacle à leur esprit créatif.

c) Il ne semble pas évident d'utiliser une méthodologie unique pour tous les projets du fait des caractéristiques propres qu'ils présentent. Nous pouvons en partie justifier ce problème en examinant l'historique de la méthodologie de la CGER.
Lorsque le M.A.I. a été créé, la méthodologie préconisée était la méthodologie ISM (Information System Management) d'IBM; celle-ci s'appliquait à tout le développement du projet jusqu'à et y compris l'analyse technique. Selon les témoignages recueillis, cette méthodologie, ne correspondant pas aux besoins, n'était pratiquement pas utilisée. Dès lors, le service M.A.I. a eu notamment pour fonction de remodeler cette démarche afin de fournir une méthodologie sur mesure, conforme aux besoins. Ces besoins ont été dégagés sur base de deux grands projets. L'inconvénient de ceci est que la méthodologie résultante a été fortement influencée par la philosophie de ces deux projets; par ailleurs, une solution apportée à un besoin exprimé par le premier ne convenait pas nécessairement au même type de besoin chez le second. A fortiori, nous pouvons conclure que cette méthodologie est difficilement applicable telle quelle à d'autres projets.

d) Le manque de motivation des développeurs pour utiliser une méthodologie peut s'expliquer par le fait qu'ils ne perçoivent pas la rentabilité éventuellement apportée par celle-ci. Cette rentabilité peut être considérée, d'une part, par l'aspect gain à la maintenance et, d'autre part, par l'aspect accroissement de la qualité du système développé. Ces deux aspects doivent être évalués par rapport au temps consacré à l'application de la méthodologie. Malheureusement, jusqu'à présent il n'existe aucune méthode satisfaisante qui permette d'évaluer cette rentabilité. Il faut remarquer que ce problème de non-motivation sera également présent avec une méthodologie moins stricte.

e) Dans cette liste d'arguments, interviennent également des contingences de type organisationnel. En effet, imposer une méthodologie stricte nécessite la mise en oeuvre d'un important système de contrôle. Pour y parvenir, l'idéal serait d'adjoindre un responsable des méthodes à chaque équipe de développement, ce qui constituerait une charge importante au niveau des ressources humaines.

5.2. DESCRIPTION DU TYPE DE DEMARCHE ADOPTEE PAR LES DEVELOPPEURS

5.2.1. LA PORTEE DE NOS OBSERVATIONS

Rappelons que la méthodologie que nous avons décrite ci-dessus a été développée il y a plusieurs années en vue de répondre aux besoins des développeurs. Ces besoins ont été dégagés sur base de deux projets pilotes.

Aujourd'hui, nous pouvons faire le point sur l'impact de cette méthodologie dans le développement de projets. Nous avons donc abandonné le terrain méthodologique pour celui du développement afin de mieux percevoir le mode de travail adopté lors d'un développment de projet.

Notre démarche ne doit pas être considérée comme une enquête car, le temps manquant, nous n'avons pas respecté les règles qui s'imposent (choix d'un échantillon représentif, élaboration d'un questionnaire "soigné"...). Nous nous efforcerons donc de ne pas tirer de conclusions hâtives au départ de ces observations, notre but étant uniquement d'apporter un éclairage nouveau à notre étude.

5.2.2. LA DEMARCHE ADOPTEE PAR LES DEVELOPPEURS

Nous avons observé que si la méthodologie stricte est rarement appliquée, en revanche, l'analyse effectuée dans le cadre du développement de projet procède d'une démarche méthodologique. Ainsi, tous les principes sous-jacents aux méthodologies universellement reconnues sont respectés (fil conducteur, utilisation d'un ensemble de modèles, établissement d'une documentation appropriée comprenant une justification des décisions prises, forte implication des utilisateurs,...).

Nos observations sont les suivantes :

La demande de développement est souvent remplie par les utilisateurs afin que l'instance de décision puisse l'approuver. Ce document est obligatoire pour un grand nombre de projets car, pour ces derniers, il constitue une condition sine qua non à l'obtention d'un code d'application.

Une fois la demande approuvée, les développeurs rédigent la définition de projet qui reprend principalement les informations définies pour cette phase dans la méthodologie. Cependant, ces informations sont rédigées en texte libre et sont souvent plus détaillées. La faiblesse de cette phase se situe peut-être à l'étape des solutions possibles puisque seule la solution retenue fait l'objet d'une documentation en langue naturelle. Les diagrammes général et détaillé tâches-utilisateurs qui sont proposés par la méthodologie ne sont pas utilisés mais leurs informations sont reprises dans le texte libre. Le document recueillant toutes ces informations est souvent appelé "cahier des charges".

En ce qui concerne l'analyse conceptuelle des données et des traitements, elle semble être la phase la plus constante pour les différents projets développés.
En effet, les données sont reprises dans un modèle Entité/Relation et les traitements sont décomposés en tâches et activités qui font l'objet d'une description. Suite à nos observations, ces descriptions sont réalisées en texte libre, en pseudo-code, à l'aide de table de décision, par la définition de règles de traitement et/ou en langage algébrique.
Il faut cependant ajouter que même si la décomposition d'un projet en traitements plus simples est réalisée, cela ne veut pas nécessairement dire que cette décomposition respecte les critères issus des définitions de la tâche et de l'activité, définitions qui ne sont pas toujours bien comprises par les développeurs. Par conséquent, ils procèdent à une décomposition selon leur logique propre et, ensuite, ils collent une étiquette tâche ou activité selon le niveau du traitement dans la découpe.

Il faut souligner que l'analyse conceptuelle est suivie de près par les utilisateurs, ces derniers peuvent ainsi apprécier les données et les traitements indépendamment de tout choix d'implémentation. La section des méthodes, quant à elle, assure un certain contrôle au niveau des données qui consiste essentiellement à vérifier la sémantique du schéma ainsi que les choix de normalisation.

La consolidation (tâches-entités et tâches-utilisateurs) n'est pas réalisée systématiquement, elle résulte des confrontations développeurs/utilisateurs et développeurs/responsables de la méthodologie. Cette dernière confrontation est cependant assez délicate car il faudrait que le groupe des méthodes participe à toute la démarche de développement, ce qui est difficilement envisageable vu le nombre de projets qui sont développés. Les développeurs étant conscients de ce problème, ajoutent dans toute leur documentation les arguments qui ont déterminé leur choix. On pourrait cependant se demander si ces arguments sont toujours convaincants.

La réalisation de l'analyse fonctionnelle est un peu plus floue.
En effet, la quantification des données pose certains problèmes aux développeurs. D'une part, ils n'en apprécient pas toujours la portée et, d'autre part, ces quantifications sont parfois difficiles à évaluer. Cependant, elles sont exigées par la section administration des bases de données qui est chargée d'établir la description de la base de données. Et, dans certains cas, on observera une collaboration étroite entre les développeurs et cette section.
Au niveau des traitements, on procédera à la description des écrans. Un dossier d'utilisation du système est également rédigé sur base de ces écrans, du modèle conceptuel des données et de la décomposition des traitements précédemment réalisés, un dossier d'utilisation du système est rédigé. Cette tâche incombe aux développeurs ou aux utilisateurs (testeurs) selon leurs compétences respectives. En effet, certains développeurs, de par leurs expériences professionnelles, maîtrisent le domaine d'application du projet et, inversément, certains utilisateurs, possèdent un bagage informatique suffisant pour rédiger ce dossier.

Le modèle physique des données est ensuite réalisé par la section administration des bases de données avec éventuellement la collaboration des développeurs.
Parallèlement, des spécifications techniques des traitements sont fournies aux programmeurs. Il ne nous est pas possible d'en décrire le contenu car il dépend de nombreux paramètres (projet, analyste, programmeur, solution adoptée,...).

La programmation, les tests et la mise en production sont les dernières étapes du développement.

En particulier, les tests d'application (validation du logiciel en fonction des spécifications) sont effectués par les développeurs ou les utilisateurs selon leurs compétences comme pour la rédaction du dossier d'utilisation du système.

5.2.3. AVANTAGES DE CE TYPE DE DEMARCHE

Nous appellerons ce type de démarche, démarche libre par opposition à l'application d'une méthodologie stricte. Nous avons essayé de dégager les avantages de cette démarche.

a) Le degré de liberté laissé aux développeurs leur permet de mettre au point un projet au moyen de la démarche qui leur est la plus familière.

b) L'absence de règles précises permet aux utilisateurs de laisser libre cours à leur créativité.

c) Chaque démarche est adaptée aux caractéristiques du projet.

Une démarche libre ne résout cependant pas le manque de motivation de la part des développeurs puisque cette motivation est fonction de la rentabilité de la démarche. Néanmoins, l'avantage d'une démarche libre par rapport à une méthodologie stricte permet aux concepteurs de ne pas aborder toutes les phases préconisées. En effet, sur base de son projet, un développeur peut juger que certaines phases ne sont pas primordiales dans le sens où elles ne contribuent pas suffisamment à la rentabilité de l'analyse.

5.2.4. INCONVENIENTS DE CE TYPE DE DEMARCHE

Une démarche libre ne résout pas tous les problèmes, il faut donc rester conscient de ses inconvénients.

a) Le danger de donner une liberté totale pour le développement est que certains concepteurs moins consciencieux, pressés par l'échéance, risquent de ne pas envisager les étapes indispensables pour le développement d'un projet.

b) Dans la majorité des cas, les responsables de la méthodologie ne peuvent suivre tout le processus de développement. Ils peuvent tout au plus servir de conseillers aux développeurs pour certaines phases critiques. Ainsi, s'ils ne sont pas complètement

impliqués dans le processus de développement, leurs interventions risquent de ne pas toujours être efficaces. Ce n'est que s'ils s'impliquent dans le projet en se documentant sur le déroulement de toute l'analyse qu'ils seront aptes à donner les meilleurs conseils .

c) La démarche libre autorise le concepteur à choisir ses propres modèles. La non-intervention des responsables des méthodes lors de la conception risque de laisser dans la documentation des schémas mal documentés et sujets à interprétation.

d) L'objectif d'aide à la maintenance attaché à une démarche de développement suppose que la documentation qui l'accompagne réponde à des critères d'homogénéité dans la manière dont les différents aspects sont traités. Il est souhaitable que la personne responsable de la maintenance puisse savoir où elle pourra trouver les informations qui lui sont nécessaires sans être pour autant obligée de consulter toute la documentation. Or, comme les démarches sont différentes d'un projet à l'autre, il n'existe aucun consensus quant à la manière dont il faut ordonnancer les différentes informations dégagées lors du développement (classification par objectif, par niveau d'abstraction,...).

6. LES POINTS DE PASSAGE

Le groupe méthodologie du service M.A.I. est aujourd'hui conscient qu'une méthodologie stricte n'est pas applicable. Par ailleurs, comme nous venons de le voir, laisser une totale liberté aux développeurs pose un certain nombre de problèmes. Ces deux facteurs conjugués vont donner naissance à une nouvelle tactique qui fera fonction de compromis entre ces deux extrêmes.

Cette tactique consiste en la définition de "points de passage". Un point de passage est caractérisé par une documentation prouvant qu'une étape du développement a bien été envisagée. Un point de passage peut être indicatif ou bloquant suivant qu'il est souhaitable ou imposé mais, dans les deux cas, il n'y a pas de contrôle de la véracité des informations contenues dans la documentation. Tout point de passage est lié aux objectifs attachés à l'étape correspondante. Le rôle de la méthodologie est de proposer aux développeurs un éventail de techniques lui permettant d'élaborer la documentation relative à ces points de passage.

Ce procédé permettra de conférer plus d'homogénéité entre les mêmes phases de développement pour des projets différents tout en laissant un certain degré de liberté aux développeurs.

Pour ce, le prix à payer est la grande diversification des modèles qui seront utilisés pour répondre aux objectifs associés à une étape de développement. Les développeurs seront alors contraints d'utiliser cette démarche mais, avec le degré de liberté qui leur est laissé, nous supposons que ceci sera envisageable dans la pratique.

Nous illustrerons cette nouvelle tactique en exposant les points de passage bloquants que le groupe méthodologie propose. Parmi ceux-ci, certains ont déjà été élaborés, d'autres sont encore à l'état de projet.

6.1. LES POINTS DE PASSAGE ETABLIS

a) La demande de développement telle qu'elle a été exposée au point 2.1.1. du chapitre 1.

b) La validation du modèle conceptuel des données. Cette validation résulte d'une discussion du schéma entre les responsables de la méthodologie et les développeurs du projet. Remarquons que, vu le nombre

restreint de personnes affectées à la section des méthodes, cette discussion se limite le plus souvent à la vérification de la bonne utilisation de la sémantique du modèle.

c) le passage en production.

6.2. LES POINTS DE PASSAGE A L'ETAT DE PROJET

6.2.1. LA DEFINITION DE PROJET

Le M.A.I. reconsidère actuellement le type d'informations qu'elle devrait contenir. Malheureusement, nous ne sommes pas encore en mesure d'exposer en quoi elle consistera car la priorité est donnée au point de passage suivant.

6.2.2. PALOMA (PAth LOad MAtrix).

"Le but de Paloma est double" c'est-à-dire, d'une part, "la réalisation du modèle dynamique des données, lequel sert à l'élaboration du modèle logique des bases de données" et, d'autre part, "l'estimation de la charge représentée par une tâche, ce qui permet au développeur de comparer les coûts de différentes tâches.

PALOMA est un point de passage dans la mesure où il oblige les développeurs à procéder à la décomposition du système en activités et tâches.

a) Les informations nécessaires

Paloma doit disposer de certaines informations pour pouvoir être utilisé. Ces informations sont :
- le modèle conceptuel des données quantifié que nous avons appelé modèle statique au point 2.3.3.b) du chapitre 1,
- la décomposition du système en activité et tâches,
- pour toutes les tâches, le nombre maximum d'exécutions par jour et la description des différents accès ponctuels aux données (transactions). La description comprend le pourcentage de données que chaque accès doit fournir par rapport au nombre total de données de même type.

b) Les informations fournies

Paloma fournit 3 types de rapports :

- le premier reprend pour chaque tâche le coût journalier de chacune de ses transactions. Ce coût représente le nombre de fois qu'un chemin attaché à la transaction est parcouru. Ce coût peut être ajusté en fonction de pondérations attribuées selon le type d'accès (lecture, écriture,...);
- le second reprend pour chaque chemin, le nombre de fois qu'il est parcouru par jour (sur base des transactions de toutes les tâches).
- le troisième est un récapitulatif des accès au niveau de tout le système. C'est à partir de ce récapitulatif que l'on pourra déduire un modèle logique des accès.

Parmi tous ces points de passage, la validation du modèle conceptuel des données et la mise en production font l'objet d'un contrôle plus approfondi du contenu informationnel de la documentation.

7. SYNTHESE

Dans ce chapitre, nous avons tout d'abord décrit la méthodologie que le service M.A.I. a élaboré pour le développement de projets. Ensuite, nous l'avons comparée avec l'analyse fonctionnelle développée à l'Institut d'Informatique de Namur. Cette comparaison nous a semblé intéressante étant donné que nous avions l'habitude d'utiliser l'analyse fonctionnelle lors de la réalisation de nos travaux pratiques à l'Institut. D'un point de vue purement théorique, nous n'avons pas constaté de différences fondamentales dans les principes de ces deux méthodologies. En d'autres termes, ce ne sont pas les divergences que nous avons dégagées qui permettent d'expliquer les problèmes d'applicabilité auxquels la méthodologie de la CGER doit faire face. De ce fait, lors de notre stage, nous avons voulu mettre en évidence, sur base de quelques observations, les causes sous-jacentes à ces probèmes. Nous avons constaté que pour définir une méthodologie stricte applicable dans un environnement, il faut tenir compte de cet environnement (utilisateurs de la méthodologie, cadre organisationnel du processus de développement,...).

Il serait intéressant de se demander dans quelle mesure une méthodologie stricte serait applicable si nous disposions de moyens coercitifs adéquats. Nous ne pouvons apporter qu'une réponse partielle à cette question sur base de notre expérience à l'Institut. En effet, à l'Institut l'obligation d'utiliser une méthodologie stricte dans un groupe de travail n'a pas supprimé pour autant tous les problèmes. Malgré que nous travaillions en groupes restreints, nous avons été confrontés à des difficultés liées à l'utilisation de modèles (choix de représentation du réel perçu, interprétations différentes, limitations des représentations auxquelles il faut se conformer,...). Dans la suite de notre étude, nous traiterons spécifiquement ce type de problèmes.

Dans l'état actuel des choses, nul ne peut nier que l'application d'une méthodologie demande un investissement de temps considérable sans pour autant apporter les résultats escomptés. En effet, les développeurs préfèrent passer du temps à chercher des solutions adéquates plutôt que de suivre une méthodologie à la lettre. Et donc, conscients que les problèmes dus à l'environnement ne sont pas solubles à court terme, les responsables de la méthodologie proposent une solution intermédiaire : les points de passage. Actuellement, nous n'avons pu énoncer que des avantages théoriques dont l'évaluation ne sera possible que lors de la mise en oeuvre de ce type de démarche. En tout cas, ce système tel que nous l'avons décrit présente des lacunes non négligeables. D'une part, le contrôle du contenu informationnel de la documentation associée à chaque point de passage n'est actuellement prévu que pour deux de ceux-ci, à savoir la validation du schéma conceptuel des données et la mise en production. D'autre part, l'aspect traitement ne bénificiera d'un point de passage que par l'intermédiaire de PALOMA.

CHAPITRE 2

LES OUTILS D'AIDE A LA DOCUMENTATION ET AU DEVELOPPEMENT DE PROJETS INFORMATIQUES

1. INTRODUCTION

Face à la crise du logiciel, de nouveaux outils ont été mis au point; le plus souvent, ils sont destinés à assister le développeur aussi bien dans la conception que dans la documentation de projets informatiques. On parlera, en général, d'outils d'aide au développement de logiciels. Ces outils couvrent une ou plusieurs phases du cycle de vie d'un projet (analyse, codage, tests, maintenance,...) et ont pour objectif d'augmenter la productivité tout en améliorant la qualité du produit.

Mais qui dit aide au développement et aide à la documentation dit définition de standards, de règles, de techniques et de méthodes.

Ainsi, l'utilisation de ces logiciels nécessite que l'on accorde une attention particulière à la définition d'une démarche méthodologique ainsi qu'à l'administration des données qui seront manipulées.

Nous ne reviendrons pas sur l'aspect méthode étant donné qu'il a déjà été présenté dans le premier chapitre.

L'aspect administration des données, quant à lui, sera abordé dans ce chapitre; nous en énoncerons les grands principes qui sont suffisants à la compréhension du rôle des outils. Nous reconnaissons que cette fonction mérite une étude beaucoup plus approfondie mais celle-ci dépasserait le cadre de notre mémoire.

En ce qui concerne les outils, nous aborderons tout d'abord la famille des dictionnaires des données qui, de par leur ampleur, sont souvent considérés comme les plus importants. Nous décrirons de manière assez détaillée en quoi ils consistent; puis, nous illustrerons nos propos par un bref compte rendu des brochures du constructeur de DATAMANAGER (le dictionnaire des données utilisé à la CGER).

Dans un deuxième temps, nous ferons un tour d'horizon des autres types d'outils d'aide au développement.

Toutes les descriptions qui suivent ont été inspirées d'articles et/ou de brochures. Nous n'avons pas l'intention ici d'évaluer ces outils les uns par rapport aux autres, ni d'étudier s'ils répondent bien à leurs objectifs mais bien de décrire tous les concepts sur lesquels notre étude va se baser. Néanmoins, ça et là, nous avancerons déjà quelques critiques et problèmes en fonction du contexte.

2. LA FONCTION D'ADMINISTRATION DES DONNEES

2.1. NAISSANCE DE LA FONCTION

Il y a plus de quinze années, beaucoup de programmes d'application travaillaient encore avec leurs propres fichiers. Si deux programmes voulaient utiliser le même fichier, ce dernier devait être dupliqué. Ceci supposait que les modifications apportées dans l'un des fichiers devaient être répercutées dans le second. L'utilisation de plusieurs fichiers posait ainsi d'énormes problèmes de redondance d'information et de confidentialité. De plus, l'accroissement du nombre des applications risquait à court terme d'entraîner des incohérences entre les données exploitées. Sont alors apparues les bases de données auxquelles les programmes peuvent accéder de manière partagée. Cette centralisation des informations apporte beaucoup d'avantages dont la diminution de la redondance, la maintenance de la confidentialité et la facilité d'utilisation des données par les programmes. [GRAAS, (c)]

Conjointement, est apparue la fonction d'administration de bases de données.

"Ses domaines de responsabilité sont :

- la description de la base de données;
- le contrôle de l'accès aux données (par la définition de privilèges d'accès sélectifs pour les programmes utilisateurs);
- la garantie du bon fonctionnement de la base de données (en veillant à ce que son utilisation en soit optimale);
- la protection de la base de données (en vue d'empêcher tout accès aux données, autre que les accès prévus par le système lui-même);
- la mise au point de la base de données (en ce compris les réorganisations)." [PEETERS, p 55]

D'autres rôles sont également imputés à la fonction d'administration de bases de données. En effet, dans une entreprise, il est fréquent que différents départements développent des applications selon leurs propres besoins. Ainsi, chacun attribue ses propres définitions aux données manipulées. La conséquence de ceci est que, bien souvent, une même donnée se voit attribuer des noms et des définitions différentes (synonymes) ou encore qu'un même nom est attribué à des données différentes (polysémie) [CANNING, p 4]. L'administration des bases de données doit permettre l'élimination de ces ambiguïtés.

2.2. EVOLUTION DE LA FONCTION

Dans la majorité des entreprises, l'information prend une place de plus en plus importante au point d'être considérée comme un capital au même titre que les ressources financières, matérielles ou humaines. Pour que ce capital puisse être géré comme les autres ressources, il faut avoir les moyens de le maîtriser, de le contrôler. L'information, c'est aussi la connaissance des ressources technologiques, c'est-à-dire de tous les éléments relatifs à la capacité informatique de l'entreprise. Aussi, la fonction d'administration de bases de données, après avoir été orientée SGBD, prendra désormais en considération des éléments comme, par exemple, des traitements, des utilisateurs, des unités physiques... Son rôle sera de maintenir les descriptions de toutes ces données et de gérer leur cohérence. Cette fonction est souvent appelée "Information Resource Management" [DOLK, KIRSCH, p 48] mais, par la suite, nous y ferons référence par les termes : administration des données.

Remarquons que très souvent le contrôle de ces ressources technologiques est limité au niveau opérationnel du système d'information; quelques tentatives ont été effectuées dans le but d'assurer également la gestion de la documentation établie durant tout le cycle de vie des différents projets. Il faut veiller à sa cohérence, à sa tenue à jour, à sa localisation, etc. En effet, cette documentation constituée d'un ensemble de dossiers est aussi considérée comme une donnée précieuse à l'organisation tant pour les développeurs que pour les utilisateurs.

Selon [HAINAUT (a), p 3], "l'administration des données doit assurer quatre types de fonctions :

- gestion, surveillance et contrôle du système d'information (1);
- aide à l'extension et à l'automatisation du S.I.;
- élaboration, contrôle et support de standards liés à la production, à la transmission et au traitement des informations;
- accueil des utilisateurs.

On observe que ses rôles sont en relation directe avec les fonctions de conceptions du S.I.

---

(1) Un système d'information est l'ensemble des composants d'une organisation relatifs aux ressources en information."

2.3. MODES D'EXECUTION DE LA FONCTION

Cette fonction centralisée pourrait être réalisée manuellement mais l'utilisation d'un système automatisé permet d'en faciliter certains aspects voire d'élargir leur portée. De ce fait, certaines entreprises ont décidé de développer un logiciel sur mesure (in-house development) en vue d'atteindre cet objectif . Il semble cependant que cette démarche ne soit pas conseillée "du fait de la difficulté de maintenance due, d'une part, à la mobilité du personnel et, d'autre part, aux demandes successives du département de traitement des informations." [CANNING, p 10]

Par ailleurs, plusieurs constructeurs ayant pressenti ce besoin commun auprès des utilisateurs ont lancé sur le marché un certain nombre d'outils susceptibles de répondre aux aspirations des entreprises dans ce domaine. Ces outils ont été initialement baptisés "Data Dictionary/ Directory". Ces premiers systèmes visaient surtout une approche d'administration des bases des données. Par la suite, ils ont été étendus, pour la plupart, à l'administration des ressources informatiques. Aujourd'hui donc, par référence à la fonction d'"Information Resource Management", ces outils sont quelquefois appelés "Information Resource Dictionary Systems" [DOLK, KIRSCH, p 48]. Pour notre part, nous conserverons la terminologie initiale abrégée "DD" pour Dictionnaire des Données.

Nous décrirons tout d'abord le concept de DD pris à la base tel que lancé sur le marché il y a une dizaine d'années; ensuite nous examinerons brièvement une des perspectives envisagées pour ce type d'outil.

3. L'OUTIL PRINCIPAL : LE DICTIONNAIRE DES DONNEES

3.1. DEFINITION D'UN DICTIONNAIRE DES DONNEES

Des multiples définitions se trouvant dans la littérature, nous avons déduit les idées suivantes en vue d'obtenir la définition la plus complète possible.

Un dictionnaire des données est un système d'information composé :

- d'une base de données dont les éléments contiennent des informations au sujet de concepts utilisés lors du développement et de la maintenance de projets informatiques (données, traitements, utilisateurs, ressources matérielles,...). Ces données au sujet de données ou META-DONNEES peuvent être mises en relation entre elles comme dans toute base de données;
- de fonctions de manipulation et d'interrogation du dictionnaire;
- de fonctions d'exploitation de ces informations;
- de fonctions de gestion de la base de données assurant l'accès partagé, la sécurité, l'intégrité et la récupération;
- d'interfaces fonctionnelles avec d'autres logiciels.

3.2. OBJECTIFS D'UN DICTIONNAIRE DES DONNEES

Idéalement, un dictionnaire des données devrait pouvoir résoudre un certain nombre de problèmes liés à la documentation; cependant, les limites relatives à l'intégration de cet outil sont souvent sous-estimées et, de ce fait, les objectifs qui lui sont habituellement assignés ne sont pas toujours atteints dans la pratique.

Le premier objectif d'un DD est de servir de support à la fonction d'administration des données au sens large (c'est-à-dire "information resource management"). La poursuite et la réalisation satisfaisante de cet objectif dépend donc de la capacité qu'a l'organisation de poursuivre les objectifs assignés à la fonction d'administration des données. D'après [HAINAUT (a), passim], peu d'entreprises peuvent se vanter d'avoir pu implanter une fonction performante d'administration des données. La mise en oeuvre d'une telle fonction se confronte à des problèmes de type organisationnel, politique ou financier. Notons également que les investissements requis incluent des personnes, des compétences, des responsabilités et des moyens informatiques et que les gains ne sont que différés et quelquefois difficilement prévisibles.

Un second objectif assigné au DD est de permettre à l'entreprise d'étendre la fonction d'administration des données, notamment par le biais d'interfaces avec d'autres logiciels. Ceci devrait permettre d'augmenter la productivité du département informatique et la qualité des projets développés.

Un dernier objectif est de faire servir le DD de support à la méthodologie, d'une part, comme outil de documentation et, d'autre part, comme système d'activation automatique de la démarche méthodologique.

3.3. STRUCTURE DE LA BASE DE DONNEES D'UN DICTIONNAIRE DES DONNEES

Cette structure est comparable à celle de toute base de données traditionnelle. Elle se compose d'entités, de relations et d'attributs qui sont attachés à ces entités et relations. Le DD est souvent fourni avec une description bien précise de sa base de données; néanmoins, rappelons que de nombreux fournisseurs prévoient des facilités qui permettent à l'organisation de modifier et/ou d'étendre (sous certaines contraintes) la description de cette base de données suivant ses propres besoins.

La portée du dictionnaire s'étend à trois niveaux d'abstraction. Tout d'abord, nous les décrirons brièvement les uns par rapport aux autres, ensuite, en vue de donner une idée bien précise du mode d'utilisation le plus répandu de ces niveaux dans une entreprise, nous tenterons de les illustrer par un cas particulier simple.
Nous reconnaissons que le vocabulaire utilisé est assez lourd mais, selon nous, il représente bien les relations existant entre les différents niveaux; une autre terminologie aurait pu prêter à confusion.

Ces niveaux sont :

- le niveau des "méta-méta-données"

  Il correspond à la description de la base des données du dictionnaire. Les "méta-méta-données" sont sous-jacentes à cette description, elles ont chacune leur propre type et ont pour fonction d'enregistrer des informations au sujet de leurs occurrences, les éléments de la base de données du dictionnaire.

- le niveau des "méta-données"

  Ces "méta-données" sont les éléments de la base de données du dictionnaire. Elles sont donc regroupées par type et ont généralement un identifiant dont la portée

s'étend à toutes les méta-données du dictionnaire. Leur rôle est d'enregistrer des informations au sujet de "données".

- le niveau des "données"

La relation existant entre une "donnée" et le dictionnaire réside dans le fait qu'il existe dans le dictionnaire une méta-donnée la décrivant.

3.3.1. CAS PARTICULIER

Nous examinerons ici comment on peut utiliser un dictionnaire des données en vue de documenter des descriptions de fichiers. Ainsi, on voudrait enregistrer pour chaque fichier une série d'informations comme, par exemple : une spécification du fichier rédigée en texte libre, la spécification et le format de chaque type d'article qu'il renferme, sa clé primaire, etc.

a) Le niveau des méta-méta-données

A ce niveau, représentons le schéma conceptuel sous-jacent à la description de la partie de la base de données relative à notre cas particulier.

Les règles sémantiques du modèle Entité/Association sont respectées telles que définies dans [BODART (a)]. Cependant, nous devons ajouter la contrainte suivante au schéma : la portée de l'unicité des identifiants doit s'étendre aux deux types d'entités que nous allons définir, i.e. deux entités de type différent ne peuvent avoir la même valeur pour leur identifiant.

Le type d'entité enregistrement représente la classe des descriptions de fichiers à propos desquelles on voudrait enregistrer de l'information.
Remarquons que nous avons choisi de considérer la clé primaire comme attribut d'un enregistrement.

Le type d'entité élément représente la classe des types d'articles qui appartiennent aux descriptions des fichiers.
L'association constituant fait correspondre à une occurrence d'enregistrement (ou d'élément) 0 ou plusieurs occurrences d'éléments (ou d'enregistrements). Nous avons choisi "0" pour le nombre minimum de la connectivité pour assurer une certaine souplesse d'utilisation du dictionnaire (il doit être possible à l'utilisateur d'introduire une description d'enregistrement alors que les éléments constitutifs de ce dernier n'ont pas encore été enregistrés).

b) Le niveau des méta-données

Les méta-données introduites dans le dictionnaire traduisent les occurrences des types d'entités et des types de relations définis dans le schéma conceptuel des données du dictionnaire.

En d'autres mots, une méta-donnée de type enregistrement décrit un enregistrement particulier et contient également la liste des identifiants des éléments qui lui sont associés (mode de représentation de la relation). De même, une méta-donnée de type élément décrit un élément particulier et contient la liste des identifiants des enregistrements dont il fait partie.

Méta-données de type ENREGISTREMENT

1. Identifiant : EN0001
   Nom : CLIENT
   Description : "..."
   :
   Clé primaire : EL0001
   Contient : EL0001, EL0003, EL0004

2. Identifiant : EN0002
   Nom : FOURNISSEUR
   Description : "..."
   :
   Clé primaire : EL0002
   Contient : EL0002,EL0003, EL0004

Méta-données de type ELEMENT

1. Identifiant : EL0001
   Nom : N°CLIENT
   Description : "..."
   :
   Format : NUM 12
   Fait partie de : EN0001

2. Identifiant : EL0002
   Nom : N°FOURN
   Description : "..."
   :
   Format : ALPHANUM 15
   Fait partie de : EN0002

3. Identifiant : EL0003
   Nom : NOM
   Description : "..."
   :
   Format : CHAR 20
   Fait partie de : EN0001, EN0002

4. Identifiant : EL0004
   Nom : PRENOM
   Description : "..."
   :
   Format : CHAR 20
   Fait partie de : EN0001, EN0002

Ces méta-données ont ici pour fonction de décrire des données opérationnelles.

Remarquons que tous les champs de ces méta-données sont complétés. Dans la réalité, lors de la description de l'enregistrement de type CLIENT, il se peut que les éléments le constituant n'aient pas encore été définis dans le dictionnaire. Notons donc, pour mémoire, que les DD offrent différents mécanismes destinés à faciliter la mise à jour des champs définissant les relations; nous l'illustrerons dans la description de DATAMANAGER (cfr. 3.7.2.).

c) Le niveau des données décrites par les méta-données

Voici quelques exemples de données décrites par les méta-données définies ci-dessus.
Nous rappelons que ces données ne font pas partie du dictionnaire mais bien des fichiers que nous venons de décrire.

Records de type CLIENT

1. N°CLIENT : 215
   NOM : Van Pevenaeyge
   PRENOM : Jan

2. N°CLIENT : 1027
   NOM : Durand
   PRENOM : Pierre

Records de type FOURNISSEUR

1. N°FOURN : W5
   NOM : Willems
   PRENOM : Luc

2. N°FOURN : A4
   NOM : André
   PRENOM : Marc

3.3.2. SYNTHESE

En conclusion, la base de données d'un DD contient des méta-données groupées par type qui décrivent les objets du système d'information. Habituellement, on retrouve les catégories de types suivantes : les données, les traitements, les utilisateurs et quelquefois les équipements physiques. La catégorie des données, par exemple, comprend notamment les types : base de données, sous-schéma, fichier, enregistrement, élément,...
Pour chaque type de méta-données sont définis des attributs communs à tous les types (identifiant, nom, description,...) et des attributs propres à ce type. Les relations ont, quant à elles, un type qui sera déterminé par le type des méta-données sur lesquelles elles sont définies.

"Les relations définies entre ces entités (méta-données) peuvent être structurelles ou sémantiques (associatives).

Les relations structurelles sont de trois natures :
- les relations analytiques (ou explosives) permettent, pour une entité donnée, de connaître de quelles entités elle est composée;
- les relations synthétiques (ou implosives) permettent, pour une entité donnée, de connaître de quelles entités elle fait partie;
- les relations intra-structurelles sont des relations établies entre les entités qui ne peuvent être associées par des relations de type analytique ou synthétique.

Les relations sémantiques, quant à elles, ne sont pas explicites, elles se déduisent de l'analyse du contenu des entités. Ainsi, les entités peuvent être regroupées en catégories ou familles sur base de mots-clés qui leur sont associés." [JANSSEN, p 626]

Remarque : Rappelons qu'à l'origine, les DD ont été conçus comme support à la fonction d'administration de bases de données. Ceci a limité longtemps la structure du schéma conceptuel des données pour la plupart des dictionnaires. Par la suite, cette structure a évolué

de manière à élargir le domaine des données que l'on veut voir documentées par le dictionnaire. Ainsi, naissent de nouveaux types d'entités destinés à décrire par exemple un document, un modèle, des règles de traitement,...

A ce sujet, nous devons signaler que cette classification en types risque de poser un certain nombre de problèmes lorsque l'on s'attache à en déterminer toutes les propriétés d'une manière générale.

3.4. CRITERES DE CLASSIFICATION DES DICTIONNAIRES DE DONNEES

Il existe différents types de DD sur le marché. Dans ce point, nous expliciterons quelques critères permettant d'établir une classification de ces dictionnaires. Pour la plupart de ces critères, nous avons tenté de définir les états extrêmes. Il est rare qu'un dictionnaire vérifie un de ces états, il faut donc considérer qu'entre ces extrêmes existe toute une gamme d'états intermédiaires. C'est alors via une combinaison de l'état intermédiaire auquel il répond pour chaque critère qu'un dictionnaire pourra être classé. L'ordre dans lequel nous aborderons les critères est déterminé par l'ordre croissant de leur degré de technicité.

3.4.1. STRUCTURE FERMEE, OUVERTE OU EXTENSIBLE

"Un dictionnaire des données a une structure fermée s'il est livré "prêt à l'emploi". Tous les éléments y sont prédéfinis (les objets, les relations, leurs attributs) ainsi que le mode d'utilisation du dictionnaire. Toute modification, par exemple l'ajout d'un attribut, s'avère très difficile et très coûteuse.

Un dictionnaire à structure ouverte est un dictionnaire mettant à la disposition de l'utilisateur un système de gestion de base de données. L'utilisateur doit définir lui-même les membres de la base de données avant de commencer l'exploitation du dictionnaire.

Un dictionnaire à structure extensible est un juste milieu entre ces deux extrêmes. Certains objets, relations et attributs sont prédéfinis mais des fonctions d'ajout ou de suppression d'éléments sont prévues." [WALTER, pp.II 1-9, II 1-11] Ceci permet à l'organisation d'adapter le DD à ses propres besoins. Ces extensions peuvent être réalisées de deux manières : soit par l'acquisition d'un nouvel ensemble optionnel d'objets, soit par la définition de nouveaux types (qui doivent souvent respecter les contraintes établies par la structure des types initiaux).

Apparemment, ce type de dictionnaire serait le plus logique à utiliser dans la pratique quoique la redéfinition de nouveaux membres ne soit pas toujours une opération aisée car elle peut, dans certains cas, entraîner des incohérences dans le DD si les contrôles adéquats ne sont pas effectués.

3.4.2. DISTINCTION DICTIONNAIRE - DIRECTORY

Dans la majorité des cas, les DD sont constitués de ces deux parties :

"La partie dictionnaire a pour fonction de décrire les ressources informatiques existantes en en donnant leur signification et leur structure logique.

La partie directory décrit des ressources sous leur aspect physique (par exemple, description des privilèges d'accès pour une donnée).
Si l'on considère les ressources de type "donnée", dictionnaire et directory sont respectivement analogues à la description logique et physique d'une donnée." [DOLK, KIRSCH, p 49]

Ainsi, "ces méta-données concernent l'homme et la machine :

- la fonction dictionnaire fournit des définitions sources à l'usage des utilisateurs d'un recueil de données;
- la fonction directory fournit des définitions à l'usage de la machine (cfr. point 3.5.2. a) génération de la Data Division en Cobol qui est une fonction Directory);

ici, "à l'usage de" indique aussi bien une fonction d'aide qu'une fonction de direction." [LUYCKX, p. 9]

Par la suite, nous considérerons un dictionnaire des données comme étant un système disposant de ces deux fonctions.

3.4.3. TYPES D'INTERFACE AVEC D'AUTRES LOGICIELS

Le type des interfaces existant entre le DD et d'autres logiciels détermine son degré d'intégration avec ces logiciels. Par conséquent, cette notion est relative; néanmoins, "un DD sera qualifié de dictionnaire intégré s'il constitue la seule source d'informations automa-

tisée dans le système." [ALLEN, p 246] Ce type de dictionnaire est aussi appelé dictionnaire actif par opposition au dictionnaire passif ou stand-alone. Notons qu'un tel niveau d'intégration est rarement atteint dans la pratique.

a) Interfaces statiques

Une interface statique entre un DD et un logiciel permet à ce dernier d'accéder aux informations du dictionnaire de manière indirecte. En effet, les informations dont le logiciel a besoin sont stockées dans un fichier intermédiaire qu'il pourra exploiter. Ce fichier peut contenir, par exemple, des descriptions formatées de fichiers pour un programme Cobol.

Ces interfaces sont assez souples mais leur inconvénient est qu'elles ne peuvent répercuter immédiatement dans les logiciels les modifications apportées aux méta-données du DD. Il faut donc utiliser des systèmes de contrôle adéquats en vue de conserver la cohérence des méta-données.

b) Interfaces dynamiques

Une interface dynamique fournit au logiciel un accès direct au DD. Cette interface est souvent réalisée par l'utilisation d'un ensemble de commandes de haut niveau destinées à activer des fonctions du DD. Un programme utilisateur pourra ainsi interroger le DD par des appels et les méta-données fournies comme résultat seront toujours à jour.

Par ailleurs, les logiciels ayant une interface avec le DD peuvent mettre à jour des informations dans le dictionnaire. Ce type d'interface permet d'assurer une cohérence des méta-données. Cependant, "il est rare que tous les logiciels intéressés par les informations se trouvant dans le dictionnaire disposent d'une interface dynamique. Une telle interface nécessite que toute demande d'information passe par le DD, ce qui risque de poser d'énormes problèmes de performance. De ce fait, un compromis est souvent adopté." [BOLK, KIRSCH, p 49] Un autre inconvénient de ce type d'interface est qu'il peut entraîner des problèmes de concurrence d'accès à la base de données du dictionnaire.

3.4.4. DEPENDANCE PAR RAPPORT A L'ENVIRONNEMENT

Par environnement, il faut entendre un SGBD, une configuration hardware, un système d'exploitation,...
Nous nous limiterons ici à examiner la dépendance d'un DD vis-à-vis des SGBD. Cette dépendance est mutuelle : elle est fonction, d'une part, du degré selon lequel le

DD repose sur un SGBD pour les fonctions de management (accès aux données, contrôle de la concurrence, sécurité, restart/recovery,...) et, d'autre part, par la source que le SGBD consulte pour obtenir les méta-données lui permettant d'effectuer des accès à ses propres bases de données.

Nous examinerons successivement trois approches partant d'un DD indépendant pour aboutir à un DD totalement intégré à un SGBD.

a) Le dictionnaire indépendant ou free-standing

Ce dictionnaire est autonome et ne repose sur aucun SGBD. Il assure lui-même les fonctions de management pour sa propre base de données. Ce type de dictionnaire peut être utilisé dans des environnements très divers composés de 0, 1 ou plusieurs SGBD différents qui disposent chacun de leur propre source de descriptions de données. Si le DD est relié à un SGDB par une interface, cette dernière a souvent une fonction limitée à la génération de descriptions de données (en DDL) pour le SGBD.

b) Le dictionnaire dépendant

Dans ce cas, le dictionnaire est considéré par le SGBD comme toute autre base de données. C'est donc le SGBD qui assure les fonctions de management pour le dictionnaire qui en est dépendant. Le DD contient les méta-données concernant la structure des différentes bases de données du SGBD mais ce dernier dispose toujours d'une version de ses propres descriptions de données. L'interface entre ces deux systèmes peut être dynamique ou statique suivant éventuellement que les produits proviennent ou non du même constructeur.

c) Le dictionnaire intégré

Dans ce cas, le dictionnaire est un composant du SGBD. Il constitue la seule source de méta-données pour ce système. Le SGBD doit donc consulter le DD pour toute transaction sur une de ses bases de données; il peut, par exemple, extraire du dictionnaire un sous-schéma d'une base de données ou encore un ensemble de contraintes d'intégrité qu'il doit vérifier avant d'effectuer telle ou telle transaction. Ce type de dictionnaire est intéressant si l'environnement n'est composé que d'un seul SGBD.

3.5. LES PRINCIPALES FONCTIONS D'UN DICTIONNAIRE DES DONNEES

Dans ce point, nous décrirons les fonctions techniques communément reconnues aux DD. Nous en envisagerons un éventail aussi large que possible et tel que, pour chaque fonction décrite, il existe sur le marché un dictionnaire qui soit capable de la réaliser (car, comme nous l'avons vu, il existe différents types de DD sur le marché).

Parmi ces fonctions, certaines sont destinées à supporter la tâche de l'administrateur des données qui peut, grâce au dictionnaire, constituer un catalogue d'informations relatives aux données qu'il doit gérer.

D'autres fonctions ont été créées en vue d'étendre le champ d'action de l'administrateur des données, fonctions qui n'auraient pu être envisagées sans les capacités techniques d'un outil comme le dictionnaire; citons, par exemple, les contrôles automatiques de cohérence et la transmission d'informations centralisées vers d'autres logiciels.

Notre objectif n'est pas de distinguer par la suite ces deux classes de fonctions. Une telle distinction serait intéressante à établir mais elle nécessite une définition de l'administration des données plus précise que celle que nous avons donnée jusqu'à présent. Malheureusement, à travers la littérature relative à ce sujet, nous avons remarqué que tous les auteurs ne sont pas unanimes quant au sens qu'il faut attribuer à cette fonction. De ce fait, nous pensons qu'il vaut mieux laisser à chacun le soin d'établir la distinction, s'il le désire, et ce, sur base de sa propre conception de la fonction d'administration des données.

Notre objectif est donc ici uniquement de fournir au lecteur une idée des opérations fonctionnelles que l'on peut effectuer avec un dictionnaire.

3.5.1. MANIPULATION, INTERROGATION ET DIFFUSION DES DESCRIPTIONS

Le DD est composé d'une base de données comprenant la description des composants d'un système d'information. A cette BD est associé un langage de manipulation et d'interrogation de ces descriptions. Le DD pourra ainsi fournir des réponses à des questions du type :

- combien de programmes seront affectés par la modification de la longueur de tel champ?
- quel(s) département(s) utilise(nt) tel fichier et à

quelle fréquence?
- quelle est la définition de tel composant du système d'information et qui en est responsable?
- quels sont tous les fichiers utilisés par tel système?
- quels sont les éléments décrits dans le dictionnaire susceptibles d'être synonymes de tel composant du système d'information? (Cette fonction a pour paramètres des mots-clés relatifs au composant)

Ces questions peuvent être posées de manière interactive ou, éventuellement, via des appels émanant de programmes d'application.

Le DD fournit également un langage de commandes d'édition de rapports sur base d'ensembles d'informations sélectionnées dans la BD. Ces rapports sont soit standard, soit définis par le responsable du DD via l'attribution de valeurs à certains paramètres. Remarquons que, dans certains cas, le DD peut ranger les résultats d'une commande dans un fichier séquentiel exploitable par un programme d'application. Ceci présente un intérêt pour le cas particulier des rapports qui peuvent, de cette manière, être définis et générés par le programme. L'administrateur des données pourra disposer de documents formatés selon ses propres besoins qui lui permettront d'effectuer différentes analyses sur les données. Citons, à titre d'exemple, des tableaux synthétiques à double entrée (données,traitements), des listes traduisant la hiérarchie d'appel des programmes, des schémas conceptuels, etc.

3.5.2. GENERATION DE DESCRIPTIONS EXPLOITABLES PAR DES LOGICIELS

Nous illustrerons cette fonction par deux exemples repris parmi les plus fréquemment rencontrés.

a) Génération d'éléments de la Data Division en Cobol

Si le DD contient des informations au sujet du type des enregistrements d'un fichier Cobol, il peut générer une description du fichier qui sera exploitable par le compilateur du langage (FD dans la File Section).
Ceci allège la tâche du programmeur qui peut, dans son programme d'application, remplacer les descriptions de fichiers par des commandes de génération adressées au dictionnaire.

b) Génération de sous-schémas de bases de données

Pour le SGBD de type hiérarchique IMS, par exemple, chaque programme se voit attribuer ce que l'on appelle un "Program Specification Block" (PSB). Un PSB contient

entre autres, la description de la vue que le programme peut avoir sur chacune des bases de données auxquelles il doit accéder. Cette vue (ou Program Control Block) est un ensemble de descriptions de segments et de champs utiles au programme et auxquelles est associée la liste des opérations autorisées (mise à jour, suppression,...)

Certains dictionnaires, par exploitation des informations qu'ils renferment, sont capables de générer ces PSB dans le langage adéquat de description des données (DDL).

3.5.3. INTERFACES AVEC D'AUTRES LOGICIELS

Les DD qui, au départ, étaient quelquefois vus comme de simples systèmes d'aide à la documentation évoluent aujourd'hui en véritables systèmes d'administration des données. Ceci explique l'installation de nombreuses interfaces entre le DD et d'autres logiciels.

Les interfaces les plus répandues couplent le DD à :

- un ou plusieurs SGBD pour lesquels on retrouve toute la gamme d'interfaces décrite au point 3.4.3.
- un moniteur TP pour que l'utilisateur puisse interroger le DD via des procédures qui lui sont familières (concept d'environnement intégré pour le dialogue homme/machine).

D'autres interfaces peuvent également être établies avec des outils d'aide au développement, de 4ème génération ou avec des ateliers logiciels. (cfr. infra point 4)

3.5.4. CHARGEMENT AUTOMATIQUE DU DICTIONNAIRE DES DONNEES

Lorsqu'une organisation prend la décision de documenter dans un DD les composants de l'entièreté de son système d'information, beaucoup de problèmes se posent. Parmi ceux-ci, celui posé par l'encodage d'un grand nombre d'informations peut être partiellement résolu avec certains DD qui fournissent une fonction de conversion. Cette fonction parcourt les programmes et les descriptions de bases de données en vue de charger les informations pertinentes dans le dictionnaire.

"Une fonction de conversion est dite équivalente à une fonction de génération si le texte source peut être regénéré à partir des informations chargées dans le dictionnaire par une fonction de conversion." [ALLEN, p 258]

Toutes ces informations doivent ensuite subir des contrôles de cohérence et de non-redondance (cfr. infra 3.5.5). Il est donc déconseillé d'utiliser la fonction de conversion sur un trop grand nombre de textes sources à la fois.

3.5.5. LES CONTROLES

Jusqu'à présent, les DD fournissent souvent des contrôles limités à la vérification de la syntaxe de la description d'une méta-donnée et à la détection de la non-unicité de son identifiant. Mais si l'on veut poursuivre l'objectif d'aide à l'administration des données avec un dictionnaire, des contrôles supplémentaires sont souhaitables. En effet, il est intéressant de disposer de procédures de contrôle de la sémantique de la description des méta-données.

Ces contrôles peuvent s'effectuer lors de l'introduction d'une description où l'on vérifierait si elle ne provoque pas d'incohérence vis-à-vis des autres descriptions. Ceci nécessite la définition de standards pour les descriptions ainsi que de règles précises de cohérence.

Ces contrôles peuvent aussi s'effectuer a posteriori sur un ensemble de méta-données ayant un point commun. Si elles appartiennent au même modèle des traitements, par exemple, on pourra détecter la non-concordance entre les sorties d'un processus et les entrées d'un autre. Des contrôles de ce type peuvent être spécifiés par les responsables de l'organisation et être ensuite greffés au dictionnaire.

3.6. PERSPECTIVES D'EVOLUTION DES DICTIONNAIRES DES DONNEES

Vraisemblablement, le DD est destiné à prendre une place de plus en plus importante dans les entreprises du fait de l'accroissement continuel du volume des informations et du caractère urgent de leur contrôle. Mais, tel qu'il existe actuellement, cet outil présente encore des lacunes : il n'est pas arrivé à un stade de maturité suffisant pour pouvoir supporter les nombreux objectifs qui lui sont assignés. Face à l'enjeu que les DD représentent dans le monde informatique, beaucoup de constructeurs et d'universités investissent dans la recherche axée sur ce type d'outils.

Dans ce point, nous décrivons une des perspectives d'évolution du DD selon laquelle il servirait de système central à un ensemble d'outils intégrés, ensemble que l'on nomme "WORKBENCH" [DIGNUM]. Chacun des outils, assigné à une des phases du cycle de vie d'un projet communiquerait avec les autres via le DD (un aperçu de ces outils sera donné au point 4.)
"Dans cette "caisse à outils", l'utilisateur peut aller puiser les outils paraissant les plus adéquats à sa tâche, et, éventuellement, les adapter et en ajouter d'autres qu'il aurait mis au point lui-même. (...) Ces outils peuvent être intégrés et présentés à l'utilisateur via un langage de commandes homogène. (...) Un bon environnement est caractérisé (notamment) par la transparence de tous ces outils à l'utilisateur. Ceci nécessite qu'un logiciel de haut niveau fasse fonction d'interface pour le dialogue homme/machine." [VAN LAMSWEERDE, pp 36,38]

La validation des résultats de toutes ces recherches demande plusieurs années car l'utilisation d'outils d'aide à la documentation et au développement de projets se heurte à des problèmes techniques, organisationnels et humains.

3.7. ILLUSTRATION : LE DICTIONNAIRE DES DONNEES "DATAMANAGER"

3.7.1. INTRODUCTION

Afin de concrétiser le concept de dictionnaire des données, nous présentons ici une description du logiciel DATAMANAGER de la firme anglaise MSP (Management and Systems Programming).

Ce dictionnaire est utilisé à la CGER depuis plusieurs années sous la responsabilité du M.A.I.

Nous avons voulu décrire à quoi ressemble cet outil lorsqu'une entreprise en fait l'acquisition et non comment il a été adapté aux besoins de la CGER. Ainsi, en considérant le dictionnaire à la base, on peut mieux distinguer les fonctions et les contrôles existants de ceux qui devront y être greffés ultérieurement.

3.7.2. DESCRIPTION DE "DATAMANAGER"

Ce produit se compose d'un logiciel de base et de plusieurs utilitaires optionnels :

a) Le logiciel de base

1. Principe

Le logiciel de base utilise une base de données comprenant six types d'entités appelés types de membres de base :
- trois types pour les traitements (SYSTEM, PROGRAM et MODULE)
- trois types pour les données (FILE, GROUP et ITEM).

Ces types ont des caractéristiques communes et des caractéristiques propres. Leur syntaxe est décrite à l'annexe B1.

L'utilisateur peut créer des membres (ou méta-données) dans le dictionnaire, un membre étant une occurrence d'un des six types de membres de base. Chacun des membres est identifié par son nom de membre ou MEMBER-NAME (chaîne de 36 caractères maximum), il peut être décrit via des clauses descriptives (cfr. infra 2.1.) et mis en relation avec d'autres membres via des clauses relationnelles (cfr. infra 2.2.). Le nom de membre ne fait pas partie de la description du membre.

2. Le langage de définition de données

2.1. Les clauses descriptives

Chaque membre peut être défini par un ensemble de clauses descriptives dépendant du type de membre auquel ce membre appartient. Une clause est composée d'un ou plusieurs mots-clés ainsi que de zones variables à remplir lors de la description du membre.

Lorsqu'il doit ajouter un membre dans le dictionnaire, DATAMANAGER peut se charger de quelques contrôles élémentaires sur ces clauses.
Exemples :
- contrôle de la longueur d'une zone,
- contrôle du respect d'un standard d'écriture,
- ...

Tous ces contrôles sont purement syntaxiques, par conséquent, on peut remarquer qu'aucune contrainte n'est imposée par le produit quant à la sémantique du contenu des clauses descriptives même si la documentation précise la signification de chacune d'entre elles.

Donc, s'il désire disposer de contrôles plus élaborés, l'utilisateur devra développer les programmes adéquats à l'aide de l'un des utilitaires optionnels : User Interface (cfr. infra b) 5.)

Afin de mieux percevoir la manière dont on peut utiliser le produit, voici quelques exemples de clauses descriptives communes à tous les types de membres :

DESCRIPTION : de type chaîne de caractères (32767 chaînes de caractères de 252 caractères).
Cette clause permet à l'utilisateur de décrire le membre comme il le désire. Lors de l'impression, les chaînes de caractères ne sont pas concaténées, ce qui permet d'éditer aisément des listes ou des tableaux.

ALIAS : de type chaîne de caractères.
Cette clause répétitive permet d'associer au membre décrit un ou plusieurs noms alternatifs ou ALIAS. L'utilisateur peut demander la concaténation du nom de membre avec un de ses alias (si cette option a été prise lors de l'installation du produit). Certains alias peuvent avoir un type prédéfini (par exemple, le nom pour l'utilisateur, le nom pour tel ou tel SGBD,...), d'autres peuvent rester quelconques. Il n'existe aucune contrainte sur la valeur de l'alias, elle peut donc être la même qu'un nom de membre, qu'un autre alias ou que toute information enregistrée dans le dictionnaire.
Des commandes spécifiques permettent d'effectuer des listes sur des alias (cfr. infra 3.2.) : liste

du ou des membres pour lesquels un alias donné a été défini, liste des alias d'un membre ou d'un ensemble de membres, liste de tous les types prédéfinis pour les alias, ...

CATALOGUE : de type chaîne de caractères.
Cette clause contient un ou plusieurs mots-clés relatifs au membre défini. Via l'utilisation d'un mot-clé, on peut aisément retrouver un membre dont on ne connait pas le nom. Par ailleurs, cette clause permet de cataloguer les membres en différentes familles (membres décrits pour un même système, par un même département, etc).
On peut ainsi définir implicitement des sous-types pour les types de membres de base (ce qui peut aussi être fait via l'utilitaire Syntaxe définie par l'utilisateur (User Defined Syntax, cfr. infra b) 8.)).
Des commandes spécifiques permettent de réaliser pour la clause CATALOGUE les mêmes opérations que pour la clause ALIAS (sauf la liste des types prédéfinis).

Remarque : la fonction de classification est plus appropriée à la clause CATALOGUE qu'à la clause ALIAS étant donné que l'on peut décrire à peu près 2000 fois plus de mots-clés que d'alias.

2.2. Les clauses relationnelles

Chaque type de membre peut être mis en relation avec d'autres via son propre ensemble de clauses relationnelles. Ces clauses se composent d'un ou plusieurs mots-clés et de zones variables destinées soit à contenir un nom de membre, soit à contenir quelques brèves informations au sujet de la relation (ces informations sont exploitables par programme; pour ce, il faut imposer des standards quant au contenu de la zone variable).

Une relation ne peut pas unir n'importe quels membres entre eux, c'est pourquoi DMR a défini un domaine d'application pour chacune des relations. Donc, pour un membre "a" de type A qui fait référence à un membre "b", DMR a défini l'ensemble des types auquel doit appartenir le type du membre "b". Le non respect de cette règle (ce qui est vérifié lors de l'encodage du membre) entraîne une incohérence au sein du dictionnaire. Ce domaine d'application ne peut être élargi mais il peut être restreint; dans ce cas, les contrôles de cohérence à l'encodage seront adaptés en conséquence.

La liste des relations ainsi que leur domaine d'application pour chaque type de membre se trouve annexe B2. Les relations y sont définies entre types

de membres, cela signifie que si les types A et B sont reliés entre eux, tout membre "a" de type A peut être relié à tout membre "b" de type B.

On peut remarquer que tout membre de type traitement (donnée) ne peut être mis en relation qu'avec des membres de types traitement (donnée) de même niveau ou de niveau inférieur. Pour certaines relations, les membres de type traitement peuvent être mis en relation avec tous les membres de types donnée (et vice versa).

Remarques :

1. Une relation réflexive au niveau des types de membres de base est aussi réflexive au niveau des occurrences de ces types de membres (appelés membres).
   Exemple : la relation CONTAINS est réflexive pour le type MODULE.

   Notation : MODULE

   Soit module "a", un membre défini sur le type de base MODULE. La réflexivité de la relation permet au module "a" d'être en relation avec lui-même via la relation CONTAINS.

2. Il n'existe pas de relation obligatoirement hiérarchique au niveau des occurrences des types de membres de base (ou membres).
   Par relation hiérarchique sur un ensemble d'éléments, on entend une relation qui donne à tout élément, un ascendant au plus et 0, 1 ou plusieurs descendants.
   Donc, un membre du dictionnaire peut, via une même relation, avoir plusieurs ascendants.

3. Le langage de manipulation des données

Les commandes décrites dans ce point ont une portée limitée au logiciel de base. Il est à noter que si certains utilitaires optionnels sont installés, la portée de ces commandes sera élargie mais cela ne sera pas décrit dans ce rapport.

Pour la bonne compréhension des commandes, il faut connaître au préalable certains aspects techniques du dictionnaire dont, notamment, ce que l'on appelle le "Source Data Set" et le "Data Entries Data Set".

Le dictionnaire est réalisé physiquement par un ensemble de descriptions codées de données reliées entre elles par des pointeurs. Toute donnée doit d'abord être décrite dans un membre source. Ce membre contient juste une description lisible par

l'utilisateur; il n'est pas encore mis en relation avec d'autres. Ce n'est que lors de l'encodage de ce membre source en membre codé que les contrôles syntaxiques s'effectueront et que les relations seront matérialisées par des pointeurs.

Les membres source sont classés dans une librairie, le "Source Data Set" (SDS), et les membres codés sont répertoriés dans le "Data Entries Data Set" (DEDS). Un membre source peut exister sans membre codé correspondant mais un membre codé ne peut exister sans membre source correspondant. Toutefois, un membre source peut être mis à jour sans que la modification ne soit directement répercutée sur le membre codé, d'où une divergence entre les deux membres.

Les commandes suivantes sont réparties en 4 classes :
- les commandes de manipulation de données,
- les commandes d'interrogation de données,
- les commandes d'édition de rapports,
- les commandes auxiliaires.

Pour la syntaxe des commandes, voir annexe B3.

3.1. Les commandes de manipulation de données

Notation : X désigne un membre source ou codé,
XS désigne un membre source,
XC désigne un membre codé,
"x" désigne un nom de membre.

INSERT
fonction : insère un membre source MS dans le Source Data Set (SDS) sauf si un membre du même nom existe déjà dans le SDS.
données : - nom du membre MS,
- description du membre MS.

ALTER
fonction : modifie un membre source MS dans le SDS sauf si le membre n'existe pas dans le SDS.
données : - nom du membre MS,
- numéro de chaque ligne à modifier ainsi que le type de modification à effectuer (insertion, mise à jour, déplacement de champs,...).
remarque : s'il existe un membre codé MC correspondant au membre source MS à modifier, la fonction conservera l'ancienne version de MS jusqu'à ce que la nouvelle version soit encodée.

COPY
fonction : insère dans le SDS comme nouveau membre source MS de nom "m", la copie d'un membre source NS de nom "n" sauf s'il existe déjà un membre source de nom "m" ou s'il n'existe pas de membre source de nom "n".
données : - le nom du membre MS,
- le nom du membre NS.

ENCODE
fonction : encode dans le DEDS, un membre MC à partir de la description du membre MS se trouvant dans le SDS
sauf : - en cas d'erreur syntaxique,
- si MS n'existe pas,
- si la description de MS prévoit une relation avec un membre dont le type n'appartient pas au domaine d'application de la relation.
données : - le nom du membre à encoder
remarques :
1. Si, dans la description de M, une relation est prévue vers un membre de nom "n" qui n'existe pas dans le DEDS, alors la fonction entraînera la création d'un membre vide, c'est-à-dire un membre sans description, de nom "n". Le type de ce membre vide sera un type par défaut suivant le type de M et celui de la relation.
2. S'il existe déjà un membre codé MC de même nom que MS, alors la description de MC sera remplacée dans le DEDS par celle de MS et les relations qui avaient été encodées pour MC seront mises à jour en fonction des relations prévues par MS.

ADD = INSERT + ENCODE

MODIFY = ALTER + ENCODE

REPLACE
fonction : donne une nouvelle description à un membre de nom "m" dans le SDS et dans le DEDS.
Nous faisons référence à la commande ENCODE pour les remarques concernant la mise à jour des relations et la création éventuelle de membres vides.
données : - le nom de membre : m,
- la description du nouveau membre.

remarques :
1. S'il existe un membre M de nom "m", cette commande revient à la séquence :
REMOVE M, ADD M.

2. S'il n'existe pas de membre de nom "m", après avoir édité un avertissement, la commande équivaut à un ADD M.

RENAME

fonction : donne un nouveau nom "m" à un membre N de nom "n" sauf s'il existe déjà un membre de nom "m" ou si des membres font référence au membre N (i.e. "n" figure dans les clauses relationnelles de leur description).

données : - ancien et nouveau noms du membre.

remarque : Dans le cas où des membres font référence au membre N, on peut réaliser la fonction RENAME par la séquence d'opérations suivante :

1. éditer la liste des membres faisant une référence à N,
2. pour tous ces membres, changer "n" en "m" dans leur(s) clause(s) relationnelle(s),
3. COPY "n" TO "m",
4. ENCODE "m",
5. REMOVE "n".

REMOVE

fonction : supprime un membre du dictionnaire ainsi que toutes les relations et les membres vides associés sauf si un membre au moins fait référence à ce membre. Dans ce cas, avant de tenter une suppression, il faut détruire ces relations.

données : - nom du membre à supprimer.

3.2. Les commandes d'interrogation de données

Les commandes d'interrogation de données et d'édition de rapports se basent sur les relations qui existent entre les membres et/ou sur la valeur de certaines clauses descriptives des membres.

Il faut donc préciser quelques concepts pour la bonne compréhension de ces commandes :

- la notion de MEMBRE VIDE
  Si, lors de l'encodage d'un membre, celui-ci doit être mis en relation avec des membres non encore codés, le système créera pour ces derniers des membres vides. Ces membres auront un type par défaut suivant la relation concernée et le type du membre que l'on encode. Ils sont identifiés dans le dictionnaire par le nom mentionné dans la clause relationnelle du membre à encoder mais ils n'ont

pas encore de description. Lorsque l'utilisateur voudra documenter le membre vide, il pourra spécifier le type qu'il veut lui donner. Si ce type n'est pas le type par défaut et qu'il est valide pour la relation, le membre vide aura le type choisi par l'utilisateur et pourra être complété.

- la notion de relation DIRECTE ou INDIRECTE au niveau des occurrences des types de membres de base du dictionnaire.
  Une relation directe est une relation entre un membre et son ascendant ou son descendant immédiat. Une relation indirecte relie deux membres via au moins deux relations directes ascendantes ou descendantes consécutives.

Nous pouvons à présent décrire les commandes :

DOES
fonctions : 1. indique si un membre M est ou non descendant direct d'un membre N sauf si N n'est pas un membre codé ou si N est un membre vide.
2. indique si un membre M est ou non descendant indirect d'un membre N sauf s'il n'est pas un membre codé ou si N est un membre vide.
données : - nom des membres M et N.

SHOW
fonction : indique quels sont les types d'alias spécifiques prédéfinis ainsi que le nombre d'alias généraux que l'on peut utiliser pour décrire tout membre.

WHAT
fonctions :
WHAT USES : indique les membres qui soit sont descendants directs uniquement, soit sont descendants indirects du membre M.
WHAT CONSTITUTES : indique les membres qui soit sont ascendants directs uniquement, soit sont ascendants indirects du membre M.
WHAT IS : indique si un libellé est un nom de membre, un alias ou un mot-clé.
WHAT FORMS : indique quels sont les membres répertoriés par un mot-clé ou par une combinaison de mots-clés. Cette commande peut être combinée avec les commandes WHAT USES et WHAT CONSTITUTES ainsi qu'avec un ensemble de conditions plus précises auxquelles doivent satisfaire les membres.

Exemples de conditions :
- le membre doit faire référence (doit être référencé) directement (ou indirectement) via une clause relationnelle déterminée.
- le membre doit avoir des attributs spécifiques qui répondent à des conditions particulières.

données : spécifier les données et les conditions suivant le type de commande utilisé.

WHICH

fonctions : 1. mêmes fonctions que pour WHAT mais sur une sélection plus précise d'un groupe de membres (sélection sur base d'une partie du nom de membre, de sa date d'introduction ou de modification, de l'utilisateur qui l'a introduit, etc).
2. indique quels alias et/ou mots-clés se rapportent à un membre M (directement ou via ses descendants).

données : spécifier les critères de sélection.

WHOSE

fonction : donne la liste des noms des membres correspondant à un alias donné.

donnée : spécifier l'alias.

remarque : cette commande est utile lorsque l'utilisateur ne connaît un membre que par un de ses alias.

3.3. Commandes d'édition de rapports

PRINT

fonction : imprime une partie ou la totalité d'un membre source ou de plusieurs membres sources sauf si le(s) membre(s) n'existe(nt) pas dans le SDS.

données : - nom du ou des membres,
- mention éventuelle des lignes à imprimer.

LIST

fonction : imprime les membres d'une ou plusieurs catégories (cfr. WHICH) avec éventuellement des informations concernant les alias. Cette fonction peut s'appliquer aux membres sources.

données : spécifier les critères de sélection.

REPORT

fonction : imprime des informations au sujet d'un ou plusieurs membres codés plus éventuellement des informations au sujet des membres qui en sont descendants.

données : - nom du ou des membres pour lesquels on veut des informations,
- éventuellement, le type de membre jusqu'où l'on veut descendre dans la hiérarchie. Ce type doit être inférieur hiérarchiquement au type du membre objet de la commande.

GLOSSARY
fonction : imprime pour les membres d'une ou plusieurs catégories (cfr. WHICH) des informations concernant certaines de leurs clauses descriptives et/ou leurs alias et/ou des informations au sujet des relations de ou vers ces membres.
données : spécifier les critères de sélection.

FORMAT
fonction : permet à l'utilisateur d'éditer un rapport (commmande REPORT) selon un layout parmi les layouts proposés. (cfr. b) 4. l'utilitaire Output défini par l'utilisateur (User Defined Output) pour les layouts composés par l'utilisateur).

Remarques
La commande PRINT s'applique à des membres source alors que la commande REPORT s'applique à des membres codés.
La commande LIST permet de répertorier des membres codés et les membres sources correpondants en vue de déceler une divergence éventuelle (dans le cas où une mise à jour d'un membre source n'aurait pas été répercutée sur le membre codé correspondant).

3.4. Les commandes auxiliaires

PERFORM
fonction : assure l'exécution d'une ou de plusieurs commandes sur un ensemble sélectionné de membres

BULK
fonction : assure l'exécution d'une commande ENCODE, PRINT ou REPORT sur un ensemble sélectionné de membres.

KEEP
fonction : accumule dans la mémoire centrale des informations résultant de commandes BULK, GLOSSARY, LIST, PERFORM, WHAT, WHICH ou WHOSE. Ces informations sont destinées à

être traitées via une des commandes BULK, GLOSSARY, LIST, PERFORM et WHICH.

ALSO KEEP
fonction : ajoute dans la mémoire centrale des informations (obtenues de la même façon que par KEEP) aux informations résultant de commandes KEEP ou ALSO KEEP antérieures. Cette liste sera maintenue jusqu'à l'exécution d'une commande KEEP ou d'une commande DROP adéquate.

DROP
fonction : détruit dans la liste obtenue via des commandes KEEP ou ALSO KEEP une partie des informations et ce, au moyen des commandes BULK, GLOSSARY, LIST, PERFORM, WHAT, WHICH et WHOSE.

3.5. Le système de sécurité

Dans le logiciel de base DMR, la seule protection d'accès au dictionnaire consiste en un mot de passe associé à chaque utilisateur. Si le mot de passe est valide, l'utilisateur pourra accéder à tous les membres du dictionnaire.

Pour pouvoir disposer d'un système de sécurité plus élaboré, il faut faire appel à l'utilitaire optionnel : Audit et Sécurité (cfr. infra b) 6.).

b) Utilitaires optionnels

Dans ce point, nous donnerons une description plus ou moins sommaire de chacun des utilitaires optionnels disponibles.

1. Interface avec les SGBD (Interface to Database)

Cet utilitaire permet de décrire des SGBD dans le dictionnaire.

Pour chaque interface, on dispose :
- d'un membre définissant le SGBD,
- éventuellement, d'autres types de membres selon les particularités du SGBD.

Il permet aussi la génération de descriptions de données dans le langage SGBD à partir de membres du dictionnaire (cfr. point 2).

Les SGBD pour lesquels on peut disposer d'une interface sont : Adabas, IDMS, IMS, Total, System 2000/80, Siemens Universal Data Base System.

DMR peut disposer simultanément de plusieurs interfaces différentes.
Notons que ce dictionnaire est indépendant de tout SGBD; il est capable de se charger lui-même de toutes les fonctions de management de sa propre base de données.

2. Génération de langage source (Source Language Generation)

Il permet de générer, à partir des membres DMR, des descriptions Cobol, PL/1 ou Assembleur ainsi que des descriptions dans un SGBD donné.
A l'aide de cet utilitaire, l'utilisateur peut définir le format des layouts réalisés à partir des descriptions des records.

3. Chargement automatique (Automation of Set Up)

Il permet de générer des membres DMR à partir de descriptions de records Cobol et PL/1.

4. Output défini par l'utilisateur (User Defined Output)

Il permet à l'utilisateur de redéfinir les formats des rapports initialement proposés par DMR.

5. User Interface

Il s'agit d'une interface entre le produit DMR et des programmes écrits par l'utilisateur.

1. Le programme DMR peut appeler un module écrit par l'utilisateur lors de l'exécution d'une commande. Cela permet par exemple de renforcer les contrôles syntaxiques et de prévoir des contrôles sémantiques qui seront activés lors de l'encodage d'un membre.

2. DMR peut écrire les résultats d'une commande dans un fichier séquentiel exploitable par un programme conçu par l'utilisateur (par exemple pour présenter les données dans un modèle particulier).

3. L'interface permet d'intégrer des commandes DMR dans un programme utilisateur. Les résultats de ces commandes seront placés dans une zone accessible au programme.

C'est grâce au User Interface que l'on peut greffer au DD des programmes de contrôle des données et de génération de rapports établis sur mesure.

6. Audit et Sécurité

Comme système de sécurité, le logiciel de base utilise le principe d'attribution d'un mot de passe à chaque utilisateur.

L'utilitaire Audit et Sécurité permet d'établir un système de sécurité plus élaboré sur l'ensemble des informations du dictionnaire. Il permet d'associer aux utilisateurs et aux membres des niveaux de sécurité en vue de limiter les accès ainsi que les opérations sur certains membres (insertion, modification, suppression). Chaque membre peut aussi se voir assigner un "propriétaire" (un individu ou un groupe de personnes). Dans ce cas, seuls les utilisateurs ayant des niveaux de sécurité spécifiques par rapport au degré des "propriétaires" peuvent accéder aux membres.

Par ailleurs, cet utilitaire permet l'édition de rapports d'audit retraçant toutes les opérations effectuées sur le dictionnaire.

7. L'utilitaire Status

Sa fonction est de permettre la construction de plusieurs dictionnaires logiques dans un seul dictionnaire physique.

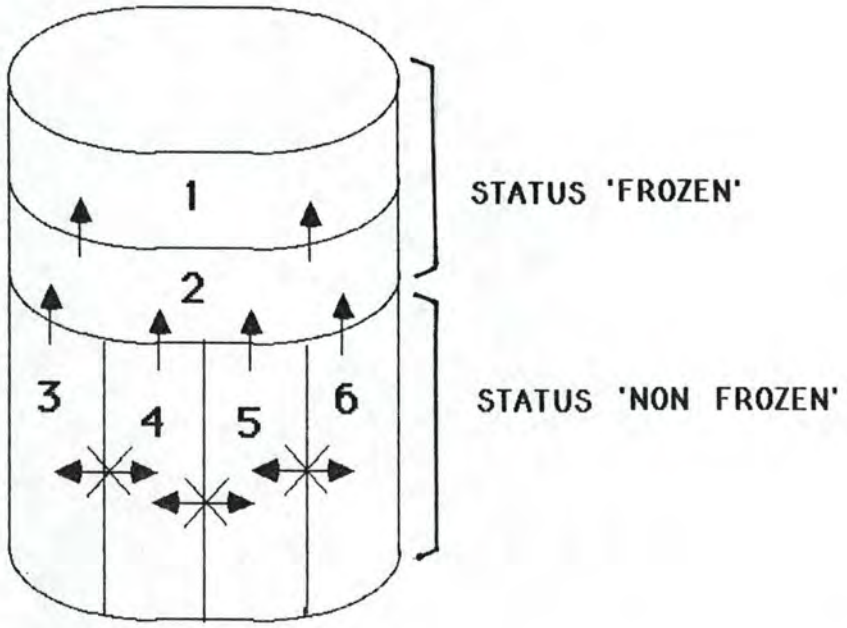
Les objectifs poursuivis sont :
- d'une part, d'enregistrer comment les membres et les relations varient à différents moments dans le temps. Pour un même membre, on peut à la fois conserver sa description périmée, courante et à l'état de projet.
- d'autre part, d'enregistrer les membres et les relations de façon à ce que les utilisateurs puissent avoir une vue différente sur les données suivant le projet, le département,etc qui les concernent.

C'est via la notion de "status" que ces objectifs peuvent être atteints. En effet, un status est associé à chaque subdivision logique du dictionnaire. Pour le premier objectif, on définira des status reflétant des phases du cycle de vie des membres (exemples : status pour tel projet, status phase conceptuelle, status implémentation).

Status "frozen" et "non-frozen".
- Un status "frozen" (gelé) contient des informations ne pouvant être modifiées.
- Un status "non-frozen" (non gelé) contient des informations pouvant être modifiées. Ce type de status convient très bien pour les phases de développement.

C'est lors de la création du dictionnaire que tous les status sont définis; à ce moment-là, ils sont dans l'état "non-frozen".
- Le responsable du dictionnaire peut rendre un status "frozen" (il gèle le status). Tous les status "frozen" sont liés entre eux par ordre chronologique de leur mise en état "frozen".
- Chaque status "non-frozen" est lié au status qui a été gelé le plus récemment.
- Il n'existe pas de relations entre les status "non-frozen".

Représentation des relations entre les status :



Le nom de membre est identifiant au sein d'un même status, ce qui permet de suivre chronologiquement l'évolution d'un membre à travers les différents status sur base de son nom de membre.

Opérations possibles dans un status "non-frozen" :
- ajouter une définition de membre à condition que ce membre n'ait pas déjà été défini dans le dictionnaire,
- modifier la définition d'un membre
  - si le membre n'existe pas alors le système émettra un message d'erreur,
  - si le membre existe mais dans un status "frozen", alors sa définition sera copiée dans le status courant pour pouvoir y être modifiée,
  - si le membre existe mais dans un status "non-frozen" alors le système émettra un

message d'erreur (toutefois, on peut remédier à la situation en copiant le membre dans le status où l'on travaille pour le modifier).

Nous examinerons sur base de la figure 1 comment la notion de status permet de donner des vues différentes sur les membres du dictionnaire.

Si l'on travaille dans un status "frozen", il est possible d'accéder aux données de ce status ainsi qu'à toutes les données des status gelés précédemment (liés par ordre chronologique) de manière à ne voir que la dernière version de chacune de ces données.
Voir vue du USER A dans l'exemple.

Si l'on travaille dans un status "non-frozen", il est possible d'accéder aux données de ce status ainsi qu'à toutes les données de l'ensemble des status "non-frozen" (liés par ordre chronologique) de manière à ne voir que la dernière version de chacune de ces données.
Voir vue du USER B et du USER C : du fait de leur status, ils voient différemment ITEM-X et GROUP-Y.

		Data Definitions Added or Modified	Seen in View from: Status 2	Status 3	Status 4
FROZEN	In Status 1	ITEM-X : ITEM HELD-AS CHAR 1 GROUP-Y : GROUP CONTAINS ITEM-1 FILE-Z : FILE CONTAINS GROUP-Y	No No Yes	No No Yes	No No Yes
FROZEN	In Status 2	ITEM-X : ITEM HELD-AS CHAR 2 GROUP-Y : GROUP CONTAINS ITEM-1, ITEM-2	Yes Yes	No No	Yes Yes
NON-FROZEN	In Status 3	ITEM-X : ITEM HELD-AS CHAR 3 GROUP-Y : GROUP CONTAINS ITEM-1, ITEM-2, ITEM-3	No No	Yes Yes	No No
NON-FROZEN	In Status 4	NEW-ITEM: ITEM HELD-AS CHAR 1	No	No	Yes
		User	A	B	C

vue sur FILE-Z : statut 1
sur ITEM-X modifié en statut 2
GROUP_Y " " " 2

vue sur FILE-Z : statut 1
sur ITEM-X modifié en statut 2
sur GROUP_Y " " " 2
sur NEW-ITEM : statut 4

vue sur FILE-Z : statut 1
sur ITEM-X modifié en statut 3
GROUP_Y " " " 3

8. Syntaxe définie par l'utilisateur User Defined Syntax (UDS)

Sans cet utilitaire, tout membre du dictionnaire doit appartenir à l'un des six types de base. Or, il serait intéressant de pouvoir disposer d'un ensemble plus élargi de types de membres, ce qui est possible avec cet utilitaire.

DMR est donc un dictionnaire des données extensible car cet utilitaire propose,
- soit de remplacer l'ensemble des six types de base par un autre ensemble de types de base (à choisir parmi trois propositions, cfr. annexe B4).
- soit de définir de nouveaux types à partir des six types de base initiaux. On aura ainsi des nouveaux types de membres de type SYSTEM, de type FILE, etc.

8.1. La définition d'un nouveau membre

a. Au niveau des clauses descriptives :

Soit le type de membre x défini à partir du type de base X.
Les clauses descriptives de x sont calquées sur celles de X, de plus, on peut associer à x de nouvelles clauses qui lui seront propres.
Pour chacune de ces clauses, on peut définir des contraintes qui devront être respectées lors des contrôles syntaxiques à l'encodage du membre.

b. Au niveau des clauses relationnelles :

Par défaut, une clause relationnelle d'un membre dont le type est redéfini sur le type de base X peut faire référence à tous les types de base auxquels le type X peut faire référence, et à tous les types de membres redéfinis sur ces derniers .

Ces relations ne peuvent être étendues mais elles peuvent être réduites lors de la description de x :
- on peut imposer que tous les types redéfinis sur un même type de base ne puissent pas être mis en relation entre eux (= non récursivité). On peut toutefois déroger à cette règle pour des relations et des types de membres spécifiques .
- on peut attribuer un niveau à chaque type de membre redéfini sur un même type de base. Ainsi, un membre d'un niveau donné ne peut faire référence à un membre de niveau supérieur défini sur le même type de base. De même, on peut déroger à cette règle pour des relations et des types de membres spécifiques.
- on peut supprimer une clause relationnelle pour un type de membre donné.
- on peut spécifier pour chaque clause relationnelle le type du membre vide qui sera créé si le membre référencé n'existe pas dans le dictionnaire.

Lors de l'encodage de tout membre, les contrôles de cohérence seront adaptés aux relations qu'aura définies le responsable du dictionnaire selon les règles exposées ci-dessus.

4. APERCU D'AUTRES TYPES D'OUTILS

Comme nous l'avons vu dans le point précédent, le dictionnaire des données peut servir de base à tout un ensemble d'outils.
Nous allons examiner ici une série d'outils d'aide au développement de projets informatiques.

Parmi ceux-ci, nous avons eu l'occasion d'en utiliser quelques-uns à la CGER, lesquels seront décrits de manière plus précise. Nous faisons référence au logiciel APE à l'aide duquel nous avons développé notre application et à l'outil QMF avec lequel nous avons créé les tables DB2 et testé les requêtes SQL.

Les autres outils seront présentés selon une typologie basée sur la ou les phases du cycle de vie auxquelles ils sont associés.
Une seconde typologie serait de distinguer les outils qui sont strictement dépendants d'une méthodologie ou d'une technique de ceux qui ne le sont pas. Nous en dirons quelques mots au travers de leur description.

Nous nous attacherons uniquement aux outils mis au point récemment (il y a moins de dix années), nous ne parlerons donc pas d'outils destinés à faciliter l'implémentation comme les librairies, les éditeurs, les préprocesseurs, les SGBD, etc.

4.1. APE - APPLICATION PROTOTYPE ENVIRONMENT

APE est un outil de développement d'applications logicielles destiné à accélérer le développement d'applications APL basées exclusivement sur des écrans ("screen-based APL applications"). [IBM (a), p.1]
Cet outil ne repose pas sur une méthodologie particulière. Néanmoins, les applications qu'il supporte sont constituées d'un certain nombre de tâches élémentaires que l'on peut regrouper en classes.

Pour satisfaire à son objectif, APE fournit, d'une part, un certain nombre de fonctions en vue de créer des "objets" nécessaires à ce type d'application [IBM (a)] et, d'autre part, un large éventail de fonctions pour manipuler les objets créés et ainsi faciliter le travail du programmeur [IBM (b)].
Il dispose aussi de sa propre "object library" et d'un outil de documentation.

Nous limiterons notre description aux fonctions de création d'objets, les autres fonctions n'étant pas essentielles à la compréhension de l'outil.

Les objets sont : les écrans, les graphiques, les déclarations de fichiers, les images, les menus, les listes et les requêtes.

4.1.1. Création d'écrans

Cette fonction permet de créer des écrans contenant du texte, des zones vides pour recevoir les données de l'utilisateur ou du système et éventuellement une zone graphique.

Le "design" d'un écran se déroule en plusieurs étapes :

- la localisation des zones,
- la définition des attributs ("output", "literal input", "numeric input", "graphics field",...) de chaque zone,
- la dénomination des zones pour pouvoir y faire référence,
- la coloration des différentes zones.

Si un contrôle syntaxique est souhaité pour les données fournies par l'utilisateur, il faut en plus pour les zones correspondantes, définir le format de la zone et le message d'erreur associé.

4.1.2. Création de graphiques

Cette fonction permet de créer des graphiques à lignes, à barres, à colonnes, à secteurs, à coordonnées polaires et des histogrammes ainsi que de leur attribuer un ensemble de caractéristiques telles que l'intitulé du graphique, le nom des axes, les légendes, les proportions, les couleurs,...
Elle permet également de spécifier les données qui doivent être représentées graphiquement. Ces données peuvent être spécifiées en valeur numérique ou sous forme d'une fonction APL exécutable.

4.1.3. Création de déclarations de fichiers

Cette fonction permet de définir le layout et le contenu d'un record d'un fichier (nom, type et longueur des différents champs d'un record) ainsi que la méthode d'accès que l'on souhaite utiliser.

4.1.4. Création d'images

Cette fonction permet de créer des sigles de compagnies ou des symboles spéciaux qui pourraient être utilisés dans le dialogue d'une application ou dans un écran.

4.1.5. Création de listes

Cette fonction permet de créer un layout de listes (nombre de colonnes, longueur des colonnes, titre des colonnes) pour recevoir un ensemble d'items. Exemple : la liste des employés d'une firme avec leur adresse et leur numéro de téléphone.
Elle permet également d'associer à une liste un ensemble d'opérations qui seraient susceptibles d'être exécutées sur les items. Exemples : mise à jour, suppression,...

4.1.6. Création de menus

Cette fonction permet de créer, dans le cadre d'une application, un ensemble de menus individuels et de les relier entre eux pour former un système de sélection de menus : à chaque option définie dans un menu, on associe une opération qui peut être soit l'affichage d'un autre menu, soit l'affichage d'un écran ou encore, un appel à une fonction. Ce système permet à l'utilisateur de naviguer dans l'application à travers les menus.

4.1.7. Création de requêtes

Cette fonction permet de créer des tables et des vues de bases de données relationnelles ainsi que de créer et d'utiliser des requêtes SQL.

4.2. QMF - QUERY MANAGEMENT FACILITY

QMF est un produit IBM classé dans les logiciels de 4ème génération. Il offre à l'utilisateur final un environnement intégré convivial lui permettant d'interroger des tables Data Base 2 ainsi que d'éditer des rapports.
Nous avons utilisé cet outil en vue de créer et de tester les requêtes SQL avant de les intégrer aux fonctions APL de notre application.

4.2.1. DEFINITION ET MANIPULATION DES DONNEES

L'utilisateur peut définir des tables DB2 de manière interactive par des primitives qu'il rédige en SQL (Structured Query Language). Il peut adjoindre à ces tables des privilèges d'accès ainsi que des index destinés à augmenter la performance des accès à la base de données.

Une fois les tables définies, elles peuvent être chargées soit par un programme spécifique indépendant de QMF, soit par des requêtes du type insertion.
Les autres requêtes de manipulation fournies par le langage sont de type mise à jour, suppression et consultation.

Pour les réaliser, deux possibilités s'offrent à l'utilisateur :

- l'utilisation d'un langage basé sur l'exemple : QBE (Query-By-Example). Avec ce langage, la structure des tables relatives à une requête est représentée directement à l'écran. On peut alors y spécifier une requête à l'aide de conventions très simples en définissant des éléments à placer dans les colonnes appropriées : labels pour le type de la requête (ex : P. pour un affichage de données), variables charnières entre plusieurs tables, expressions conditionnelles sur les données d'une ou plusieurs colonnes, etc.
  Le langage QBE est utile pour les personnes qui ne connaissent pas SQL et qui veulent effectuer des requêtes relativement simples sur une base de données.

- l'utilisation du langage SQL à l'aide duquel l'utilisateur peut définir ses propres requêtes. Pour ce, il peut disposer d'un canevas à remplir construit sur base du type de l'opération : INSERT, UPDATE ou SELECT et sur base de la ou des tables concernées par la requête.

Après exécution d'une requête de sélection, le résultat est présenté sous la forme d'une nouvelle table.

Si une requête doit être utlilisée plusieurs fois, il est possible de la sauver. Si elle est rappelée pour une exécution, l'utilisateur pourra, le cas échéant, introduire la valeur des variables à l'écran.

Remarquons qu'il n'est pas possible via QMF de rédiger des requêtes procédurales de par la philosophie d'un langage comme SQL. De ce fait, si l'on désire effectuer des traitements plus complexes sur les données, il faudra intégrer les requêtes SQL dans un programme d'application qui en exploitera les résultats.

4.2.2. EDITION DE RAPPORTS

L'utilisateur peut demander à QMF de générer des rapports comprenant des tables (données ou résultats de requêtes); il est également possible d'imprimer le texte des requêtes. Une fonction élaborée de "design" permet à l'utilisateur de formater ses rapports selon ses propres besoins.

4.3. TOUR D'HORIZON DES AUTRES OUTILS

Nous avons classé ces outils suivant la phase du cycle de vie d'un projet à laquelle ils sont associés. Certains outils peuvent couvrir plusieurs phases, dans ce cas, nous les décrirons dans le cadre de la phase à laquelle ils contribuent le mieux.

Les phases servant de base à la classification sont issues de la méthodologie proposée à la CGER telle que décrite au premier chapitre. Dans le cas où une phase n'est pas abordée, cela signifie que nous n'avons pas jugé utile de présenter les éventuels outils y afférents.

Notre objectif est ici de proposer au lescteur un aperçu des différents types d'outils (actuels ou à l'état de projet) qui sont destinés à assister l'utilisateur dans le développement de projets.

4.3.1. POUR LA PHASE DE DEFINITION DE PROJET

a) L'étude de l'existant

Nous avons vu qu'elle consiste en l'établissement de fiches des informations et des traitements existants ainsi qu'en l'élaboration de diagrammes des flux.

Il existe des outils dotés d'écrans graphiques permettant de réaliser ce type de diagramme. Ils peuvent supporter une schématisation par raffinements successifs dans le sens où des parties du diagramme global peuvent être éclatées en un nouveau diagramme plus détaillé (cfr. EXCELERATOR décrit au point b)).

Certains de ces outils se basent notamment sur la méthode d'analyse SADT (Structured Analysis Design Technique, de Softtech,inc.). Cette technique permet de décrire un système aussi bien par une vue sur les données (datagrammes) que par une vue sur les traitements (actigrammes).

Ainsi, l'on peut associer

- à une donnée : sa définition, son support, les activités qui la produisent, la contrôlent et l'utilisent;
- à un traitement : sa définition, les données en entrée, en mise à jour, en sortie ainsi que son processeur.

Chacun de ces éléments peut aussi être éclaté pour permettre une découpe par raffinements successifs.
Un tel système permet ainsi d'effectuer des contrôles de cohérence sur base de l'analyse des rapports adéquats : n'existe-t-il pas de données superflues, redondantes ou encore des données qui ne sont jamais utilisées ? etc.

Ces contrôles pourraient se faire manuellement, sur base de fiches adéquatement complétées, mais un système automatisé rendra ces contrôles moins fastidieux car il permet notamment un établissement plus aisé de références croisées.

b) Le choix d'une solution

Dans l'étude qui va suivre dans la deuxième partie de ce mémoire, nous avons voulu considérer le processus de décision comme un parcours dans un arbre, parcours dirigé par des choix intermédiaires. Ces prises de décision se font fréquemment au cours de réunions successives, il est donc nécessaire d'en garder une trace. Ceci est d'autant plus important que ce processus nécessite de temps à autres des rétroparcours dans l'arbre de décision.

[VAN LAMSWEERDE, p. 33] décrit justement un outil qui pourrait servir de support à ce processus. Ce système, au stade expérimental en 1982, associe à chaque noeud une fonction de décision multicritères. Il permet également de conserver les choix qui ont été retenus, les choix rejetés ainsi que leurs justifications et hypothèses respectives.

Une prise de décision traduisant le choix d'une solution peut également être assistée d'un Système Informatique d'Aide à la Décision, quoique ce genre de problème ne fait pas état de critères de décision précis auxquels on peut attribuer des poids de commun accord.

4.3.2. POUR LA PHASE D'ANALYSE CONCEPTUELLE

Le principal objectif ici est de pouvoir représenter le système à développer de la manière la plus précise possible.
Le processus est itératif et des outils semblent les bienvenus pour faciliter, dans une certaine mesure, les mises à jour à apporter aux descriptions et aux schémas.

Nous décrirons ici deux outils :
- IDA
- EXCELERATOR

Le premier est plutôt orienté documentation, spécification et est destiné à supporter une méthodologie tandis que le second sert de support à des techniques bien précises mais ne repose pas sur une méthodologie particulière.

Dans un dernier point, nous donnerons des indications relatives aux recherches concernant des systèmes experts qui pourraient être destinés à guider l'utilisateur dans la modélisation d'un système d'information.

a) IDA : Interactive Design Approach

Cet outil fait partie d'un atelier logiciel et est destiné à supporter la phase de définition de projet et de l'analyse conceptuelle. Il a été développé à l'Institut d'Informatique de Namur en collaboration avec le projet ISDOS développé à l'université de Michigan.
Nous avons tiré de [BODART (a), pp. 93-135] les principaux aspects de cet outil.

Ce logiciel se compose des éléments suivants :

- une base de données de spécifications rédigées à l'aide du langage de spécification DSL (Dynamic Specification Language);
- un analyseur DSA (Dynamic Specification Analyser) comprenant un module d'exécution de requêtes de mise à jour et un module de production de rapports;
- un langage interactif d'interrogation QL (Query Language) qui permet également d'effectuer des contrôles de cohérence.

"Le langage DSL fut défini pour permettre de décrire principalement les spécifications fonctionnelles d'un système d'information. Ce langage est basé sur les concepts de type d'objet, de type de relation entre ces types d'objet et de type propriété qui caractérisent les types d'objet ou de relations. Les propriétés peuvent prendre des valeurs." [BODART (b), P. 49]

Le langage DSL est extensible en ce sens qu'il permet de définir de nouveaux types d'objets.
Il est également partitionnable car il permet de grouper les objets en sous-ensembles homogènes.
"Ces sous-ensembles peuvent être associés à des "aspects" fonctionnels homogènes d'un système d'information à décrire." [BODART (b), p. 51]

La démarche méthodologique associée au logiciel suggère la liste suivante d'"aspects" :
- modèle de structuration des informations et des messages,
- modèle de structuration des traitements,

- modèle de la dynamique des traitements,
- modèle de la statique des traitements,
- modèle des ressources.

L'outil comprend également des interfaces vers d'autres logiciels spécialisés :

- un générateur automatique de simulation destiné à évaluer le caractère réalisable des spécifications fonctionnelles d'un système d'information,
- un générateur de maquettes pour tester le caractère effectif des spécifications.

Ceux-ci visent à introduire une approche plus expérimentale dans l'étude des problèmes d'organisation, d'une part, en testant l'impact de diverses hypothèses de fonctionnement et, d'autre part, en favorisant la perception directe par les utilisateurs de ce qu'ils souhaitent obtenir.

Exposons les principes de ces deux générateurs :

1. Evaluation du caractère réalisable des spécifications par génération automatique d'un programme de simulation

   Ce logiciel se base sur les spécifications dynamiques du système d'information ainsi que sur les spécifications des ressources. Par génération de programmes de simulation, on pourra rechercher une solution réalisable en procédant par ajustements successifs de certains éléments (par exemple, du seuil acceptable pour tel type de ressource, de l'enchaînement des traitements,...)

2. Evaluation du caractère effectif des spécifications par génération d'une maquette programmée du système d'information

   Ce prototype généré par application de procédures permet à l'utilisateur de prendre connaissance des spécifications du nouveau S.I. par expérimentation directe, sans devoir consulter de volumineux dossiers. L'exécution de la maquette correspond à un processus interactif au cours duquel l'utilisateur :
   - introduit des données relatives aux messages,
   - reçoit des messages l'informant de l'état des données,
   - peut demander de connaître l'état des données associées à un processus au moment du déclenchement ou de la terminaison de celui-ci,
   - est informé des raisons qui ont provoqué la terminaison d'un processus dans un état donné,
   - ...

Examinons à présent les différents types de contrôles prévus par le logiciel IDA :

1. Les contrôles effectués lors de la mise à jour des spécifications.

   Lors de l'introduction ou de la modification d'une spécification, les procédures de l'analyseur DSA procèdent à des contrôles syntaxiques et sémantiques assez limités. Les contrôles syntaxiques ne concernent que la validité syntaxique d'une relation relative à un objet donné (ex: l'unicité de l'identifiant ou la "légalité" d'une relation) alors que les contrôles sémantiques ne portent que sur l'appartenance d'une propriété à un ensemble prédéfini de valeurs.

2. Les contrôles effectués a posteriori

   Ils peuvent être exprimés via le langage d'interrogation de la base des spécifications. L'outil répond à une requête en fournissant la liste des objets vérifiant la condition associée à cette requête ainsi que la liste des objets qui ne la vérifient pas.

   D'autres contrôles peuvent être effectués par exécution de programmes particuliers réalisés selon les besoins. Citons, entre autres, les programmes de détection de circuits, de détection de synonymes et de détection de spécifications incomplètes (sur base du modèle de spécification).

b) EXCELERATOR [ABIS]

Excelerator est un outil d'aide au développement de projets informatiques qui fonctionne sur micro-ordinateur. Comme sur un produit "mainframe", chaque utilisateur a son identifiant et son mot de passe, ce qui lui permet d'instaurer des privilèges d'accès aux projets.

Le produit comprend sept utilitaires qui seront décrits ultérieurement :

- le dictionnaire Excelerator;
- l'aide à l'élaboration de graphiques;
- l'aide à la définition d'écrans et de rapports;
- l'analyse des données;
- l'utilitaire de documentation;
- l'interface avec d'autres dictionnaires;
- l'utilitaire de maintenance.

Chacun de ces utilitaires présente une interface graphique conviviale avec l'utilisateur. En effet, ce dernier évoluera dans le système au moyen de menus et il pourra réaliser les schémas à l'aide d'une souris.

Tous les utilitaires travaillent en interaction avec le dictionnaire Excelerator où sont stockées les informations concernant toutes les entités manipulées par le système, ce qui permet à Excelerator d'assurer une cohérence interne entre les données d'un même projet, notamment lors des mises à jour.

Décrivons à présent ces différents utilitaires:

1. Le dictionnaire Excelerator

   Il constitue le coeur du système Excelerator et il est accessible de tous les modules. Il peut stocker des informations (ou méta-données) au sujet des données, des relations mais aussi au sujet des écrans, des rapports, des utilisateurs,... Y sont associées également des indications du type : qui a quoi, date de la dernière mise à jour,...
   Ainsi, tous les modules d'Excelerator peuvent avoir leur propre vue sur une description unique pour un objet identifié par un même label.

   A partir de ces informations, des rapports peuvent être générés sur base de requêtes rédigées dans un langage d'interrogation; il est possible également d'en tirer des fichiers utilisables par d'autres programmes.

2. L'aide à l'élaboration de graphiques

   Cet utilitaire permet à l'utilisateur de créer différents types de schémas à l'écran.

   En effet, l'interface graphique lui permet de créer, de déplacer et d'établir aisément des objets et des liaisons entre ces objets pour la réalisation d'un schéma. Chaque objet peut ensuite être défini dans le dictionnaire Excelerator.

   Avec un tel outil, le développeur ne devra par recopier un schéma entièrement après chaque série de modifications.

   Les schémas peuvent être décrits avec plusieurs niveaux de détail : un élément de niveau i peut être éclaté en un nouveau schéma de niveau i+1. Par cette fonction, il est même possible, par exemple, d'éclater un traitement d'un diagramme des flux en un nouveau diagramme de type structuration des traitements.

   Les types de diagrammes proposés par Excelerator sont les suivants :

   - les diagrammes de flux de données pour la description des données, des traitements, des entités organisationnelles, des espaces de stockage ainsi que des flux de données.
     Deux notations sont proposées : la notation de Gane & Sarson et la notation de Yourdon.

- les schémas structurés de décomposition des traitements avec circulation des données.
  Le modèle proposé est celui de Constantine et DeMarco.

- les modèles logiques des données pour décrire des entités associées par des relations unidirectionnelles 1-1 et N-1.
  Ce modèle peut être relié à un diagramme des flux; ainsi, on pourra par exemple connaître, pour toute entité du modèle logique, la composition du fichier qui y correspond dans le diagramme des flux.

- les modèles entité/relation dans la notation de Chen ou de Merise.
  Remarquons qu'avec ce modèle, il nous semble que nous ne pourrions pas représenter un modèle entité/association tel que décrit par François Bodart. En effet, la documentation ne précise pas si l'on peut mentionner sur le schéma des connectivités ou encore des attributs à une relation.

- les diagrammes structurés (Jackson)
  Ils sont plus précis que les schémas structurés de décomposition de traitements car ils précisent en outre le type des traitements par un signe conventionnel ((°) pour une sélection, (*) pour une itération, aucun signe pour le séquencement).

- les graphes de présentation pour la représentation d'une configuration informatique, la description de l'organisation de l'entreprise,...et ce, à l'aide de tout un éventail d'icônes représentant des personnes, des terminaux, des unités à disques, des rapports, etc.

3. L'aide à la définition d'écrans et de rapports

Excelerator permet de décrire des prototypes d'écrans pour l'application ainsi que le format des rapports qui seront générés.
Remarquons à ce sujet, qu'Excelerator offre peu de moyens pour réaliser des prototypes pour les systèmes à développer; néanmoins, il est possible d'exporter vers d'autres logiciels, générateurs de prototypes, des données extraites du dictionnaire Excelerator.

4. L'analyse des données

Cet utilitaire permet de générer différents rapports (prédéfinis ou non) au sujet des informations stockées dans le dictionnaire, rapports permettant de vérifier la complétude et la cohérence de celles-ci. Il peut fournir, par exemple, la liste des connexions illégales, la liste des objets non connectés dans un diagramme des flux, des graphes de type HIPO,...

5. Les interfaces avec d'autres dictionnaires

Elles permettent le transfert de données à partir de et vers d'autres dictionnaires (Excelerator ou non).

6. L'aide à la documentation

Cet utilitaire permet d'éditer un document complet comprenant les différents graphiques et rapports du projet ainsi que des commentaires de l'utilisateur. "Ce document peut inclure des dossiers réalisés par un traitement de texte indépendant d'Excelerator, ce qui est rendu possible par une interface externe" [DIGNUM, p. 818]
S'il existe des structures standard pour certains types de document, celles-ci peuvent être stockées de façon à faciliter les rédactions futures de ce type de document.

7. L'utilitaire d'aide à la maintenance

Il permet le "back up" des opérations, la modification du nom d'un projet, des priorités d'accès, le choix d'un mode de représentation des données dans les schémas, etc.

En conclusion, remarquons que cet outil offre un grand éventail de fonctions mais il ne propose pas de méthode de travail. Pour être efficace, il faudrait qu'il soit intégré dans une démarche méthodologique.

c) Utilisation de systèmes experts

En vue d'aider l'utilisateur à établir des schémas conceptuels (entité/relation, par exemple), plusieurs types de systèmes experts sont à l'étude.
La fonction d'un tel système peut être, par exemple, d'aider l'utilisateur, par un dialogue approprié, non seulement à construire un schéma mais aussi à spécifier les informations contenues dans ce schéma, et ce, par exploitation d'une base de connaissances et d'une base de faits.

La base de connaissances contiendrait des règles traduisant la sémantique du schéma ainsi que le mode de raisonnement adopté par l'utilisateur pour la construction d'un tel schéma.

La base de faits, quant à elle, contiendrait des informations propres à l'environnement organisationnel de l'utilisateur; de plus, elle pourrait être enrichie par les informations introduites par l'utilisateur durant tout le processus de spécification.

4.3.3. POUR LA PHASE D'ANALYSE FONCTIONNELLE

Un outil comme APE peut intervenir à ce niveau pour la description des menus et écrans, il est également possible de présenter à l'utilisateur un prototype basé sur l'enchaînement de ces écrans.

Du côté des quantifications, citons le logiciel PALOMA qui a été décrit dans le cadre des points de passage pour la méthodologie de la CGER (chapitre 1, point 6). Notons que ce logiciel n'est utilisable que dans le cadre de l'application d'une méthodologie prévoyant la réalisation d'un modèle conceptuel des données et une découpe des traitements en tâches et transactions (d'après la terminologie utilisée à la CGER).

4.3.4. POUR LA PHASE D'ANALYSE TECHNIQUE

L'Institut d'Informatique a développé un système dans le cadre de l'atelier d'aide à la conception de bases de données, système destiné à assister le développeur dans la tâche de transformation de schémas.

D'une part, il s'attache à l'aide à la transformation d'un schéma entité/relation en un schéma général des accès (indépendant de tout type de SGBD).

D'autre part, il peut guider l'utilisateur à produire, à partir de ce dernier schéma, un nouveau schéma conforme à un SGBD donné. Les choix effectués par l'utilisateur sont validés par le système sur base d'un répertoire de règles de transformation de schémas et sur base de la liste des contraintes du SGBD considéré.

Notons que la production d'un schéma conforme englobe les aspects de performance dégagés sur base du type de traitements qui seront exécutés sur la base de données. Ce système nécessite donc l'utilisation d'une méthodologie bien déterminée.

4.3.5. POUR LA PHASE DE PROGRAMMATION

Au niveau des données, il existe des outils capables de générer sur base de spécifications précises, des descriptions de sous-schémas SGBD ou encore des descriptions de fichiers Cobol (Data Division).

Au niveau des traitements, "il existe des préprocesseurs puissants capables de générer une partie des programmes à partir d'une représentation de plus haut niveau, parfois même procédurale, de la solution. Les générateurs de code Cobol à partir de tables de décision en sont un exemple." [VAN LAMSWEERDE, p. 23]

La recherche se penche également sur la mise au point de systèmes experts destinés à générer des parties de code ou encore à servir de "debuggers".

Les premiers types de systèmes se basent sur le principe de programmation "synthétique" où le problème est tout d'abord spécifié dans un langage de très haut niveau et où le programme est ensuite construit par raffinements successifs. "At its most ambitious, automatic program synthesis expresses the idea that automatic programming is the purest sort of automation. A team of programmers or users would focus their efforts on developing a complete specification of a user's requirement. Then an intelligent system would write the program in such the same way that today's compilers generate machine code from high-level languages." [FRENKEL, p. 578]
Ce type de système expert doit pouvoir contenir des règles traduisant le processus de raisonnement adopté par le programmeur et les utiliser à bon escient.
Il faut remarquer que la liste des règles risque de ne jamais être exhaustive.

D'autres systèmes experts sont utilisés pour le "debugging" des programmes. A ce sujet, [FRENKEL, p.589] insiste sur le fait que "one would have to limit the interactions between software entities to those that can be analysed by the expert system. You would have to sacrifice your design freedom in order to make the expert system applicable."

Notons que les nombreux travaux effectués dans ce domaine pourraient indirectement avoir des retombées sur la compréhension du processus de conception d'un système. Ainsi, le nombre de questions que soulève l'établissement de toutes les règles utiles au système expert permettra peut-être de mettre en évidence des problèmes jusqu'alors insoupçonnés dans le développement de projets informatiques.

4.3.6. POUR LA PHASE DES TESTS

Nous ne retiendrons ici que quelques outils d'aide aux tests des programmes.

Il existe des "conducteurs de tests, qui, sur base d'un plan de test composé du code à tester, d'un ensemble de valeurs pour les entrées et les résultats attendus, exécutent le code sur les différentes valeurs en entrée et vérifient la concordance entre les résultats obtenus et les résultats attendus." [VAN LAMSWEERDE, p. 30]

Des outils plus sophistiqués sont capables de générer des jeux de test (white box) en vue d'assurer que tous

les chemins d'une procédure soient parcourus au moins une fois. [VAN LAMSWEERDE, p. 231]

4.3.7. POUR LA PHASE DE MAINTENANCE

Un certain nombre d'outils d'aide au développement incluent des fonctions d'aide à la maintenance. Celles-ci ont principalement pour rôle de fournir de la documentation dont la structure facilite les analyses. Par ailleurs, certains outils ont la capacité de conserver des informations au sujet de versions successives de données, de traitements,... de telle sorte qu'une trace des modifications peut être conservée.

5. SYNTHESE

Dans ce paragraphe, nous avons tenté de décrire quelques outils d'aide à la documentation et au développement de projets informatiques.
Remarquons qu'actuellement, il n'existe pas encore d'outils destinés à supporter de manière efficace l'ensemble des phases du développement d'un projet. Néanmoins, de nombreuses tentatives sont menées dans ce sens.

Dans notre description de ces outils, nous avons toujours fait abstraction des problèmes qui pourraient surgir lors de leur utilisation. La plupart des documents que nous avons consultés ne se limitent d'ailleurs qu'aux descriptions fonctionnelles des outils, donnant l'impression que l'intégration d'un nouvel outil dans une organisation est une opération systématique qui se déroule sans obstacles.

Or, nous nous devons insister sur le fait que la réalité est toute autre. Installer un outil demande que l'on définisse clairement les objectifs qui lui seront assignés ainsi que les éléments de l'entreprise qui seront concernés.
S'il doit constituer une aide à la documentation, il est indispensable de définir des standards ou naming conventions (s'il n'en existe pas encore) et de spécifier les informations nécessaires et suffisantes qui devront y être introduites.
S'il doit servir de support à la méthodologie, il faudra de plus déterminer quels points de la méthodologie seront supportés et de quelle manière (ceci suppose que l'organisation dispose d'une méthodologie bien définie; nous verrons dans la suite de notre étude les problèmes dus à l'instauration d'une méthodologie).
Dans les deux cas, il faudra prévoir un mode d'utilisation et un programme d'intégration de l'outil.

Ces indications ne constituent qu'un aperçu des difficultés sous-jacentes à l'installation d'un outil. Etant donné que leur origine est souvent liée à des problèmes plus graves et indépendants de l'outil en tant que tel, il est donc primordial que l'on ne néglige pas les difficultés susceptibles de survenir lors du franchissement des différentes étapes à suivre pour l'intégration d'un tel système.

DEUXIEME PARTIE

ETUDE DE L'UTILISATION D'UNE METHODOLOGIE ET D'OUTILS AU DEPART DE LA REALISATION D'UNE APPLICATION

INTRODUCTION

Après avoir étudié la méthodologie proposée à la CGER, nous avons essayé de l'éprouver en développant une application. L'objectif poursuivi était de pouvoir dégager les points faibles et les points forts de cette méthodologie.

Nous consacrerons donc le premier chapitre à la description de l'application que nous avons développée.

Nous reprendrons ensuite, dans le deuxième chapitre, les différentes étapes de développement propre à notre application pour pouvoir relever les problèmes que nous avons rencontrés en essayant de respecter la démarche méthodologique de la CGER. Nous tenterons alors dans la mesure du possible de proposer des éléments de solutions à ces problèmes dans la limite de notre application.

L'examen de ces problèmes nous permettra ensuite d'aborder l'aspect des outils. En effet, sur base des résultats obtenus, nous tenterons, dans le troisième chapitre, de donner la spécification d'outils d'aide à la documentation de projet dans le cadre d'une méthodologie. Par conséquent, nous nous sommes principalement limités à la documentation de l'application que nous avons développée.

De temps à autre, il sera fait mention de certains services de la CGER, à ce sujet, nous invitons le lecteur à se référer à l'organigramme de l'entreprise présenté dans l'introduction générale.

CHAPITRE 1

DESCRIPTION DE L'APPLICATION

1. INTRODUCTION

Le meilleur moyen de pouvoir se rendre compte des problèmes que peut poser l'utilisation d'une méthodologie pour le développement de projets informatiques est de développer une application en suivant le plus fidèlement possible la méthodologie que l'on souhaite étudier. Ainsi, nous pourrons, par la suite, dégager les points où nous nous sommes détachés de la méthodologie et essayer de comprendre pour quelles raisons.

Dans notre cas, cette méthodologie est celle proposée par la CGER que l'on retrouve dans la première partie (description de la méthodologie proposée par la CGER) et l'application c'est PUMA (Publication Managment) qui consiste à répertorier et à gérer des documents et des brochures pour le service Méthode et Administration de l'Information (M.A.I.).

Nous avons commencé le développement de l'application PUMA avec ces seules données, aussi le premier travail que nous avons dû effectuer avant de procéder à l'élaboration des différentes étapes de développement était de préciser l'application pour avoir une idée générale de sa portée, de son utilité et de son environnement.

Nous avons ensuite suivi les différentes étapes de développement. Certaines de ces étapes sont longues et leur analyse détaillée n'apporterait rien à notre étude aussi, par souci de clarté, seuls certains éléments se retrouveront dans l'exposé et ce, à titre exemplatif. Le lecteur intéressé pourra consulter les annexes pour une description et complète de l'application.

Il faut garder en mémoire que nous avons développé cette application en vue d'expérimenter la cohérence de la démarche méthodologique et de ce fait, nous avons fait abstraction des aspects performances du système.

2. PRESENTATION GENERALE

Le service Méthode et Administration de l'Information (M.A.I.) est un service de la CGER qui emploie une trentaine de personnes dont l'objectif est l'aide au développement de systèmes informatiques.

Ce service est subdivisé en plusieurs sections et chaque section contribue à sa manière à la réalisation de cet objectif (cfr. Introduction du mémoire) :
- l'administration de l'information qui s'occupe de l'aspect méthodologie,
- bases de données (IMS et DB2) et ses aspects systèmes,
- dictionnaire des données et outils de support au développement (qui s'occupe d'assurer le support au développement),
- infocentre qui s'occupe principalement de l'aspect aide aux utilisateurs,
- sécurité informatique qui s'occupe des problèmes de confidentialité et de back up,
- formation informatique pour tout ce qui concerne la gestion des cours.

Les utilisateurs de ce service sont les personnes impliquées dans le développement de systèmes informatiques, c'est-à-dire les autres services du CTI mais aussi les utilisateurs info-centre.

Mais qui dit service rendu dit échange d'informations. Cet échange se fait notamment sous forme de publications de divers types (rapport, mode d'emploi, brochure constructeur,...) qui appartiennent à l'une des 2 catégories suivantes :
- les documents rédigés dans une des deux langues nationales par le service M.A.I. ou par un autre service (documents),
- les brochures commandées par le M.A.I. auprès des constructeurs (brochures).

La distribution de ces documents et brochures doit être assurée par le M.A.I.

Actuellement, la gestion de ces publications se fait manuellement et de façon non standardisée. En effet, il existe peu ou pas de liste de distribution ni de répertorisation des publications disponibles. Plus particulièrement pour les documents en production, il est difficile de maîtriser leur cycle de vie (rédaction, dactylographie, traduction, ...).

A moyen de terme, avec le nombre des publications et des destinataires qui augmente, le service ne pourra plus supporter la distribution et la gestion y afférentes. Aussi, le but de l'application PUMA sera d'apporter une solution automatisée à ces problèmes et ce, en vue d'offrir une meilleure rentabilité dans la production des documents et dans la distribution des documents et brochures.

Nous allons maintenant détailler les différentes phases du développement de l'application PUMA en commençant par la demande de développement dont le but est d'obtenir le feu vert pour le démarrage du projet.

3. LA DEMANDE DE DEVELOPPEMENT DE PROJET

A cette occasion, un groupe de travail a été réuni avec, d'une part, l'équipe informatique, en l'occurrence nous et, d'autre part, le groupe des utilisateurs c'est-à-dire le responsable du service M.A.I. et les responsables des différentes sections.

Nous avons donc rédigé ce formulaire en étroite collaboration avec les utilisateurs (cfr. annexe C). Ces derniers ont tendance à "gonfler" les causes et origines du problème et les avantages attendus et à "minimiser" les contraintes de réalisation en vue de voir leur demande approuvée par l'instance de décision. La participation de l'équipe informatique consiste alors à tempérer les arguments des utilisateurs surtout au niveau des contraintes de réalisation.

Après avoir reçu une réponse favorable à notre demande, nous avons établi le dossier concernant la définition du projet afin de préciser les frontières, les objectifs et les contraintes du système à développer.

Mais avant d'aborder le sujet, il nous semble utile de signaler une certaine redondance entre les informations fournies lors de la demande de développement et les informations demandées dans la définition du projet (objectifs et contraintes, description du système). En fait, la définition de projet détaille ces informations à l'intention des informaticiens, des concepteurs, ... Dans la demande de développement ces informations sont destinées à aider l'instance de décision dans son choix. Les destinataires et les objectifs de ces deux documents sont donc différents.

4. LA DEFINITION DE PROJET

Les documents proposés pour supporter la définition de projet ne sont que des suggestions. Nous avons essayé de nous y conformer sans pour autant les suivre aveuglément.

4.1. L'ETUDE DE L'EXISTANT

Pour faire cette étude, nous avons procédé à des interviews auprès des personnes concernées afin de préciser la gestion actuelle des publications touchées par notre application.

Il était impossible, de prime abord, de recenser les activités et principales tâches existantes au départ de ces interviews. Ceci provient, d'une part, du fait de la gestion manuelle et très aléatoire des publications et, d'autre part du fait de la difficulté d'identifier correctement ce qu'on entendait par tâche et activité.

Pour ces raisons, nous avons décidé de rédiger cette étude de l'existant en texte libre. Par la suite, nous avons dégagé les différents "traitements" que les publications subissaient ainsi que les informations existantes afin de construire un schéma du système actuel complété par la description des différents éléments y appartenant (traitements, informations, exécutants des différents traitements,...). Envisageons ces différentes étapes.

4.1.1 ETUDE DE L'EXISTANT EN TEXTE LIBRE

Pour rappel, les publications concernées par l'application sont :
- les documents rédigés par le M.A.I. ou par un autre service,
- les brochures commandées auprès des constructeurs

et dont la distribution est assurée par le service M.A.I. Par conséquent, les documents rédigés par un autre service et dont la distribution n'incombe pas au service M.A.I. n'appartiennent pas au domaine de notre application.

La gestion des publications qui existe actuellement est différente selon qu'il s'agisse de documents ou de brochures. C'est pour cette raison que nous avons décomposé l'étude de l'existant en deux parties.

a) Les documents

Les documents sont rédigés dans une des deux langues nationales (français ou néerlandais) et traduits dans l'autre langue. La traduction est effectuée soit par le service qui a rédigé le document soit par le service de traduction (90.1). Les documents sont aussi dactylographiés et, comme pour la traduction, la dactylographie est effectuée soit par le service qui a rédigé le document soit par le pool de dactylographie. Les documents doivent être dactylographiés et disponibles dans les deux langues pour pouvoir être distribués. En vue de cette distribution, le M.A.I. se charge de faire photocopier les documents.

Tout document porte un titre. Le nom de l'auteur, le numéro de version, le nom du service, le numéro du service, la date sont parfois mentionnés.

Un ajout, un retrait et/ou une modification peuvent être apportés à un document sans pour autant qu'il subisse une mise à jour complète sauf si l'auteur le souhaite. Dans ce dernier cas, le numéro de version, s'il est mentionné sur le document, est modifié.

Les documents sont soit sur papier, soit sur support magnétique.

Il n'existe pas de liste reprenant tous les documents actuellement disponibles, cependant des listes partielles sont tenues à jour par quelques auteurs. Dans ce cas, les documents sont référencés par leur localisation physique, leur titre ou un libellé alphanumérique propre à l'auteur.

La destination d'un document est liée au type de ce document (rapport, support de cours, manuel d'utilisation,...) et au sujet traité (méthodologie, base de données,...).
Il peut être distribué
- à des groupes personnes ayant la même fonction (analystes, informaticiens A, programmeurs,...),
- aux membres d'un groupe de travail,
- aux membres d'un service de la CGER,
- à titre personnel,

ou encore à toutes les personnes du CTI ou du DSI.

Lorsqu'un document est distribué, la liste des personnes qui l'ont reçu devrait être établie. Malheureusement, c'est rarement le cas et c'est pour cette raison que la distribution des versions successives d'un même document n'est pas toujours menée à bon terme de même que pour les ajouts, les retraits et/ou les modifications d'un document.

Une "version provisoire" (cas particulier de la notion de version d'un document) n'est habituellement pas distribuée, si ce n'est à quelques personnes compétentes pour en avoir une critique.

b) Les brochures

Initialement, le service M.A.I. passe une commande aux constructeurs concernés par l'intermédiaire du service Secrétariat Informatique et Economat qui réceptionne la commande et la donne au service M.A.I. Le secrétariat dispose également de quelques exemplaires de chaque brochure pour pouvoir assurer un service d'emprunt et de consultation.

En cas de mise à jour, un exemplaire témoin est envoyé au M.A.I. qui peut, par sa consultation, décider de passer commande au constructeur responsable de la brochure mise à jour (de la même manière que pour les 1ères versions).

Les brochures portent un numéro de référence propre aux constructeurs. Ce numéro se compose d'un libellé alphanumérique et d'un numéro de version.

c) Remarques

La gestion qui nous occupe se limite aux publications (documents et brochures) prises en charge par le M.A.I. à un moment donné dans leur cycle de vie, pour être ensuite distribuées.

L'étude de l'existant décrite ci-dessus dépasse parfois la portée de notre application en ce qui concerne la gestion des documents au sein du M.A.I. En effet, le M.A.I. peut se voir attribuer des documents en provenance d'autres services à n'importe quel moment de leur cycle de vie. Or, ces publications sont reprises dans l'existant au même titre que les documents rédigés par le M.A.I. car cela devenait trop compliqué de tenir compte des deux filières.

4.1.2. REPRESENTATION DU SYSTEME ACTUEL

En vue de donner un autre éclairage du système actuel, nous avons représenté sur un schéma, un certain nombre d'éléments : les traitements, les informations qui circulent d'un traitement à l'autre, les exécutants des traitements.

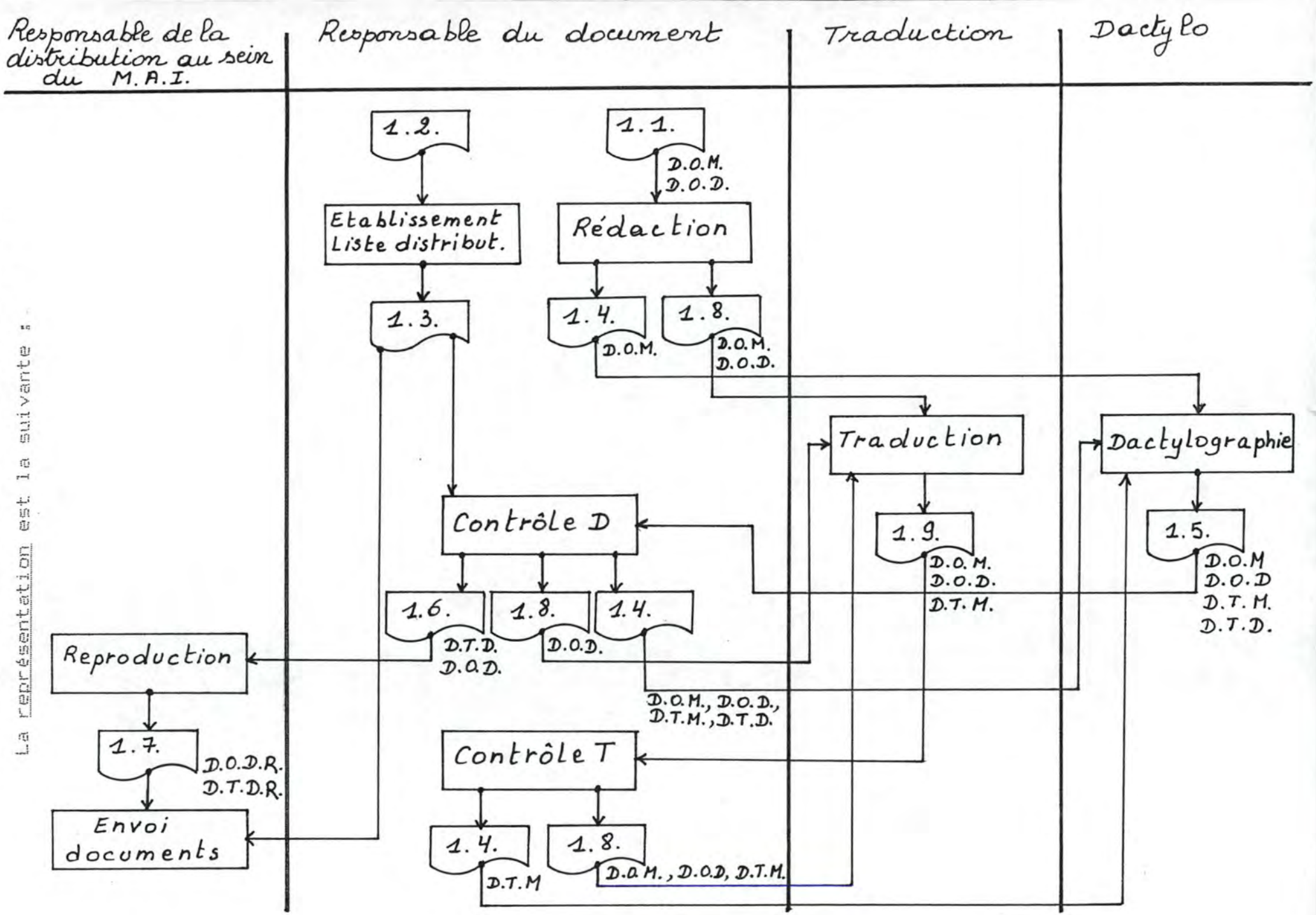
Nous avons tout d'abord dégagé deux traitements importants à savoir la gestion des documents et la gestion des brochures. Nous avons alors décidé de représenter le système actuel à l'aide de deux schémas, l'un pour les documents et l'autre pour les brochures car les traitements au sein de ces deux gestions sont assez différents.

Parmi les informations que nous avons représentées, nous avons distingué les publications des messages qui les accompagnent dans leur cheminement. En effet, les publications font partie de l'existant. En particulier, les documents subissent des modifications opérées par les différents traitements. Etant donné que nous travaillions sur le cycle de vie des documents, il nous semblait essentiel de les reprendre à ce niveau. Nous avons choisi le même formalisme pour la représentation de la gestion des brochures, quoique l'apport en information soit moindre; selon nous, le seul intérêt de reprendre les brochures dans le schéma est de pouvoir distinguer à quel moment les brochures sont prises en charge par le M.A.I.

Ce type de représentation ne doit pas être confondu avec le diagramme des flux présenté dans la méthodologie de F. Bodart parce que, d'une part, le but de notre modèle n'est pas de représenter un enchaînement chronologique des traitements et, d'autre part, nous ne travaillons pas avec la représentation des objets du réel mais bien avec les objets eux-mêmes qui dans un certain sens sont leur propre représentation.

a) Gestion des documents dans le système actuel

La représentation est la suivante :



Spécification des exécutants des traitements :

Responsable du document : personne auteur du document (ou désignée responsable si il y a plusieurs auteurs pour un document) qui se charge de sa gestion.

Traduction : soit une personne bilingue appartenant au service ayant rédigé le document (l'auteur lui-même ou un collègue), soit le service de traduction 90.1.

Dactylo : soit une personne appartenant au service ayant rédigé le document (auteur lui-même ou un collègue), soit le pool de dactylographie.

Responsable de la distribution au sein du M.A.I. : personne du service qui se charge de la reproduction des documents et de leur distribution (lorsque le document est rédigé par le M.A.I., cette personne est alors l'auteur du document).

Spécification des informations :

(D.O.M.) document original manuscrit : document manuscrit rédigé par un ou plusieurs auteurs dans une des deux langues nationales.

(D.O.D.) document original dactylographié : document simultanément rédigé et dactylographié par un ou plusieurs auteurs dans une des deux langues nationales ou document dactylographié après sa rédaction dans une des deux langues nationales.

(D.T.M.) document traduit manuscrit : document manuscrit traduit dans l'autre langue nationale.

(D.T.D.) document traduit dactylographié : document dactylographié après sa traduction dans l'autre langue nationale.

(D.O.D.R.) documents originals dactylographiés reproduits : l'ensemble des reproductions d'un D.O.D.

(D.T.D.R.) documents traduits dactylographiés reproduits : l'ensemble des reproductions d'un D.T.D.

(1.1.) demande de rédaction : message écrit ou oral justifiant la production d'un nouveau document ou la modification d'un document déjà existant.

(1.2.) demande d'établissement d'une liste de distribution : message écrit ou oral justifiant l'établissement d'une liste de distribution.

(1.3.) liste de distribution : liste des destinataires d'un document avec éventuellement le nombre d'exemplaires nécessaires pour un destinataire. Cette liste est soit écrite, soit en mémoire humaine.

(1.4.) demande de dactylo : message écrit ou oral donnant un certain nombre d'informations pour la dactylographie ou pour la correction de celle-ci.

(1.5.) fin de dactylo : message indiquant qu'un document manuscrit a été dactylographié suivant les indications fournies par 1.4.

(1.6) bon de reproduction : message écrit ou oral mentionnant le nombre d'exemplaires nécessaires pour reproduire le document.

(1.7.) fin de reproduction : message indiquant qu'un document dactylographié a été reproduit en un certain nombre d'exemplaires.

(1.8.) demande de traduction : message écrit ou oral indiquant les informations nécessaires pour la traduction ou pour la correction de celle-ci.

(1.9.) fin de traduction : message indiquant qu'un document original manuscrit ou dactylographié à été traduit suivant les indications fournies par le 1.8.

Spécification des traitements :

Rédaction : rédaction d'un nouveau document ou apport d'une modification à un document manuscrit ou dactylographié déjà existant. La rédaction peut se faire de façon manuscrite ou dactylographiée.
en entrée : éventuellement un D.O.M. ou D.O.D.
en sortie : (D.O.M. et 1.4.) ou (D.O.M. et 1.8.) ou (D.O.D. et 1.8.)
déclenchement : 1.1.

Etablissement liste distribution : création ou mise à jour d'une liste de destinataires suivant le sujet traité dans le document.
en entrée : (sujet traité dans le document)
en sortie : la liste de distribution (1.3.).
déclenchement : 1.2.
Ce traitement peut être déclenché à n'importe quel moment, cependant ce traitement doit être au moins terminé pour la phase de distribution.

Dactylographie : dactylographie d'un D.O.M. ou d'un D.T.M. à l'aide d'un traitement de texte ou d'une machine à écrire suivant les indications fournies par le 1.4.
en entrée : (D.O.M. ou D.T.M.) et 1.4.
en sortie : (D.O.M., D.O.D. et 1.5.) ou (D.T.M., D.T.D et 1.5.).
déclenchement : 1.4.

Contrôle D : vérification du D.O.D (ou D.T.D.) sur base du D.O.M. (ou D.T.M.) et, si le document est correct alors aiguillage vers la tâche suivante sinon, retour des documents à la dactylographie.
en entrée : D.O.D (ou D.T.D.) et D.O.M. (ou D.T.M.) et éventuellement 1.3.
en sortie : si correct alors (D.O.D. et 1.6.) ou (D.O.D. et 1.8.) ou (D.T.D. et 1.6.) sinon (D.O.M., D.O.D. et 1.4.) ou (D.T.M., D.T.D. et 1.4.).
déclenchement : 1.5.

Traduction : traduction d'un D.O.M. ou d'un D.O.D. dans l'autre langue nationale suivant les indications fournies par le 1.8.
en entrée : (D.O.M. ou D.O.D.) et 1.8.
en sortie : (D.O.M. ou D.O.D.), D.T.M. et 1.9. avec éventuellement le D.T.M. (pour correction).
déclenchement : 1.8.

Contrôle T : vérification du D.T.M. sur base du D.O.M. ou du D.O.D. et, si la traduction est correcte alors aiguillage vers la tâche suivante sinon, retour des documents à la traduction.
en entrée : (D.O.M. ou D.O.D.), et D.T.M.
en sortie : si correct alors (D.T.M. et 1.4.) sinon (D.O.M. ou D.O.D.), D.T.M. et 1.8.
déclenchement : 1.9.

Reproduction : reproduction d'un document en fonction du nombre d'exemplaires à produire.
en entrée : (D.O.D ou D.T.D.) et 1.6.
en sortie : (D.O.D.R. ou D.T.D.R.) et 1.7.
déclenchement : 1.6.

Envoi documents : distribution des (D.O.D.R. ou D.T.D.R.) selon la liste de distribution.
en entrée : (D.O.D.R. ou D.T.D.R.) et liste de distribution (1.3.).
sortie : rien.
déclenchement : 1.7.
Ce traitement ne peut être exécuté que si le document à distribuer existe à l'état dactylographié dans les 2 langues nationales.

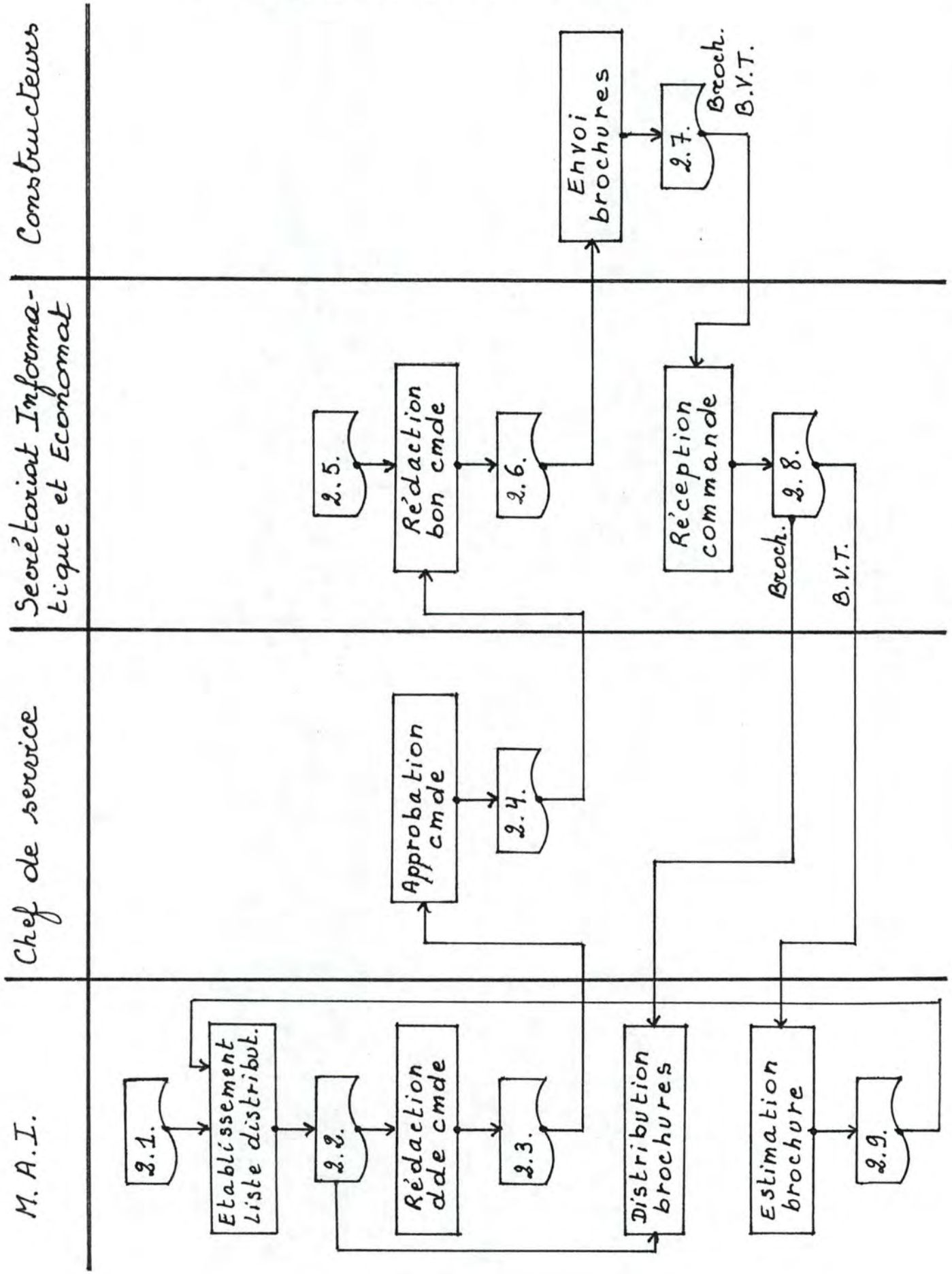
Remarques :

- certains messages sont fictifs et nous les avons ajoutés pour jouer le rôle d'événements déclenchants (exemples : les messages de fin de traduction, de fin de dactylographie ou de fin de reproduction).

- le schéma n'inclut pas de structures de contrôle (conditions, points de synchronisation,...), ce qui constitue une perte d'information quant à l'enchaînement des traitements et ce, au bénéfice d'une meilleure lisibilité du schéma. Ces informations sont cependant reprises dans la documentation annexée à la représentation.

b) Gestion des brochures dans le système actuel

La représentation est la suivante :



Spécification des exécutants des traitements :

M.A.I. : le service Méthode et Administration de l'Information

Chef de service : responsable du service M.A.I.

Secrétariat Informatique et Economat : service responsable de la passation des commandes auprès des contructeurs et la réception de celles-ci.

Constructeurs : ensemble des fournisseurs de matériel et de logiciel à la CGER.

Spécification des informations :

(broch.) brochures : ensemble des brochures commandées auprès des constructeurs.

(B.V.T.) brochure version témoin : brochure envoyée par un constructeur lorsqu'elle résulte de modifications.

(2.1.) demande de brochure : message écrit ou oral justifiant la commande d'une brochure.

(2.2.) liste de distribution : liste des destinataires d'une brochure avec éventuellement le nombre d'exemplaires nécessaires pour un destinataire. Cette liste est soit écrite, soit en mémoire humaine.

(2.3.) demande de brochure : message écrit indiquant la commande d'une nouvelle brochure avec toutes les informations la concernant.

(2.4.) demande de commande approuvée : même message que 2.3. mais approuvé par le chef de service.

(2.5.) stock épuisé : message indiquant que le stock d'une brochure est épuisé et qu'une nouvelle commande doit être faite.

(2.6.) bon de commande : message destiné aux constructeurs pour la commande de brochures.

(2.7.) bordereau de livraison de brochures : message en provenance des constructeurs pour indiquer les brochures envoyées.

(2.8.) livraison approuvée : message indiquant si les commandes faites correspondent aux livraisons de brochures et mentionnant aussi la réception d'une version témoin.

(2.9.) justificatif : message écrit ou oral indiquant si un exemplaire témoin peut ou non être commandé.

Spécification des traitements :

Etablissement liste distribution : création ou mise à jour d'une liste de destinataires suivant le sujet traité dans la brochure.
en entrée : (sujet traité dans la brochure).
en sortie : liste de distribution (2.2.).
déclenchement : sur demande (2.1.) ou si l'on éprouve le besoin de commander des exemplaires d'une version témoin envoyée par un constructeur (2.9.).

Rédaction dde cmde : rédaction d'une commande de brochure en fonction de la liste de distribution.
en entrée : liste de distribution (2.2.).
en sortie : 2.3.
déclenchement : 2.2.

Approbation cmde : étude de la demande de commande par le chef de service sur base d'une analyse budgétaire. Si cette demande est approuvée alors aiguillage vers la phase suivante (sinon, rien).
en entrée : demande de commande (2.3.).
en sortie : demande de commande approuvée (2.4.).
déclenchement : message 2.3.

Rédaction bon cmde : rédaction d'un bon de commande à destination d'un constructeur.
en entrée : demande de commande approuvée (2.4.).
en sortie : bon de commande (2.6.).
déclenchement : messages 2.4. ou 2.5.

Envoi brochures : les constructeurs envoient les brochures commandées ou une version témoin en cas de modification d'un brochure.
en entrée : bon de commande (2.6.) dans le cas des brochures commandées.
en sortie : broch. et/ou B.V.T. et un bordereau de livraison (2.7.).
déclenchement : bon de commande (2.6.).

Réception commande : contrôle de la marchandise reçue et pointage sur la liste des brochures commandées à partir du bordereau de livraison.
en entrée : broch. et/ou B.V.T., bordereau de livraison (2.7.) et bon de commande (2.6.).
en sortie : broch. et/ou B.V.T. et 2.8.
déclenchement : bordereau de livraison (2.7.).

Distribution brochures : distribution des brochures commandées selon la liste des destinataires.
en entrée : broch., 2.8. et 2.2.
en sortie : rien
déclenchement : message 2.8.

Estimation brochure : évaluation de la nécessité de commander la version témoin.
en entrée : B.V.T.
en sortie : B.V.T et 2.9.
déclenchement : 2.8.

4.1.3. CRITIQUE DE L'EXISTANT

La description de l'existant en texte libre et l'élaboration de sa représentation graphique nous ont permis de dégager la critique de l'existant. En effet, nous avons rencontré un certain nombre de difficultés lorsque nous avons voulu construire le schéma représentant la situation actuelle. Et, dans la plupart des cas, ces difficultés étaient dues aux anomalies du système. Nous avons alors décrit ces anomalies dans un rapport rédigé en texte libre que nous reprenons ci-dessous.

Les traitements se rapportant à la gestion des documents se font actuellement dans des services différents, dont le M.A.I., et la marche à suivre n'est pas toujours très précise. Ces traitements sont tout à fait manuels et très aléatoires car ils sont effectués par des personnes différentes.

Il n'existe pas de répertoire centralisé de tous les documents. Les listes de distribution sont rarement écrites, ce qui ne permet pas dans tous les cas d'assurer un suivi dans la distribution des versions successives d'un même document.

Il n'y a pas moyen de connaître facilement à quel stade de son cycle de vie se trouve un document. Cette anomalie découle du fait qu'il n'existe pas de répertoire centralisé de tous les documents.

La localisation physique des documents et des brochures n'est pas toujours facile à retrouver et il n'existe pas de numéro de référence pouvant identifier un document au sein de tout le service.

4.2. OBJECTIFS ET CONTRAINTES DU NOUVEAU SYSTEME

A présent que nous avons étudié la situation actuelle, il est temps de constater les objectifs attendus et les contraintes du nouveau système. Cette étape est essentiellement du domaine des utilisateurs aussi, comme pour l'étude de l'existant, nous avons procédé par interviews pour pouvoir les dégager.

4.2.1. LES OBJECTIFS ORGANISATIONNELS

L'objectif principal est d'améliorer le service rendu lors de la distribution des documents et brochures. D'une part, la vue centralisée de l'ensemble des informations concernant les publications, les

destinataires et les relations entre les deux pourraient rendre la distribution plus rapide et plus précise. D'autre part, la maîtrise du cycle de vie d'un document ou d'une brochure permettra d'en accélérer les différentes étapes (commandes, traduction,...).

L'objectif organisationnel découlant du premier est le gain de temps au niveau des personnes qui sont actuellement chargées de la réalisation d'une commande et/ou de la distribution de document.

4.2.2. LES OBJECTIFS INFORMATIONNELS

Ces objectifs concernent l'obtention d'informations précises et pertinentes au sujet des publications, des destinataires et des relations existant entre eux. En effet, il faut pouvoir

- effectuer le suivi d'un document (dans quel état est-il, à quel stade, ...),
- obtenir la liste de toutes les publications relatives à un thème (IMS, DB2, méthodologie,...),
- consulter, modifier, interroger la liste de distribution des publications,
- gérer la distribution d'un document ou d'une brochure en fonction de la liste de distribution,
- identifier le support (système dans lequel il est produit) d'un document,
- consulter les références complètes d'un document ou d'une brochure,
- confronter plusieurs listes de distribution.

4.2.3. CONTRAINTES

Etant donné l'organisation actuelle, il faut installer un système automatisé aussi facile que possible à utiliser. En effet, il ne doit pas constituer une charge pour les utilisateurs. Donc, le système doit être très dirigé utilisateur (user-friendly) afin qu'il puisse gérer ses propres documents et aussi consulter les différentes informations nécessaires. Le système doit être très dirigé utilisateur.

Dans les étapes suivantes, l'utilisateur peut encore préciser ses besoins et contraintes par des entretiens successifs (lors de l'élaboration des modèles des données et des traitements, du dialogue utilisateur-système,...).

4.3. SOLUTIONS POSSIBLES

Nous devions en principe faire une étude approfondie pour chacune des solutions éventuelles. Nous n'avons pas procédér de cette manière mais plutôt par éliminations successives. En effet, il n'est pas nécessaire d'étudier à fond une solution si on sait suffisamment tôt qu'elle ne conviendra pas.

Nous avons en fait réalisé une description en texte libre de la solution retenue en essayant de justifier pourquoi nous avons choisi telle solution et pourquoi nous avons rejeté les autres.

Nous pensions au départ développer un système assez contraignant où nous assurerions un ensemble de contrôles sur les enregistrements et imposer de la sorte à l'utilisateur une démarche précise pour la gestion des documents et des brochures. Il n'est pas nécessaire de faire une analyse coût-efficacité pour se rendre compte qu'il est préférable de donner le maximum de liberté à l'utilisateur. Il y a plusieurs raisons à cela :

a) un système trop contraignant peut parfois amener l'utilisateur devant un problème qu'il ne pourra résoudre qu'en modifiant le programme;
b) le peu de contraintes est un facteur jouant en faveur de la transportabilité du logiciel;
c) le risque d'oubli parmi les scénarios possibles et les contrôles y afférents.

Nous avons alors décidé de laisser la liberté aux utilisateurs afin que chacun puisse assurer la gestion des publications comme il le souhaite tout en restant aussi le seul responsable de la bonne ou mauvaise utilisation du système. Une nouvelle alternative s'offrait alors à nous : d'une part, fournir à l'utilisateur un pseudo-langage compréhensible par le système à l'aide duquel il pourra gérer les publications et interroger le système sur un certain nombre de chose et, d'autre part, fournir à l'utilisateur les fonctions nécessaires à la gestion des publications à l'aide d'un enchaînement de menus et d'écrans.

La première solution ne respectant pas la contrainte de convivialité exigée lors de l'établissement des objectifs et contraintes, nous avons opté pour la seconde pour offrir ainsi aux utilisateurs les fonctions en réponse à leurs objectifs. Le système devrait être simple à utiliser, par conséquent, il ne devrait pas requérir une longue formation pour ses utilisateurs.

La méthodologie prévoit que chaque solution soit décrite notamment à l'aide du diagrammes détaillé Tâches/ Utilisateurs. Du fait des caractéristiques de la solution que nous avons retenue, il nous a été impossible de représenter le fonctionnement du système à l'aide du diagramme proposé. En effet, c'est l'utilisateur qui détermine le fonctionnement du système via un même poste de travail et ce, par des fonctions interactives.

Il faut à présent procéder à l'analyse conceptuelle pour avoir une idée précise des données que le système aura à manipuler ainsi que des traitements qu'il supportera.

5. L'ANALYSE CONCEPTUELLE

Pour pouvoir décrire les données et les traitements, nous avons procédé à de nouveaux interviews auprès des différents membres du M.A.I. responsables de l'émission de documents et de la distribution de documents et de brochures. Il fallait, en effet, que nous ayons une idée précise des informations que nous aurions à manipuler par la suite (versions, état du document, identifications des destinataires, ...) car les traitements en dépendaient.

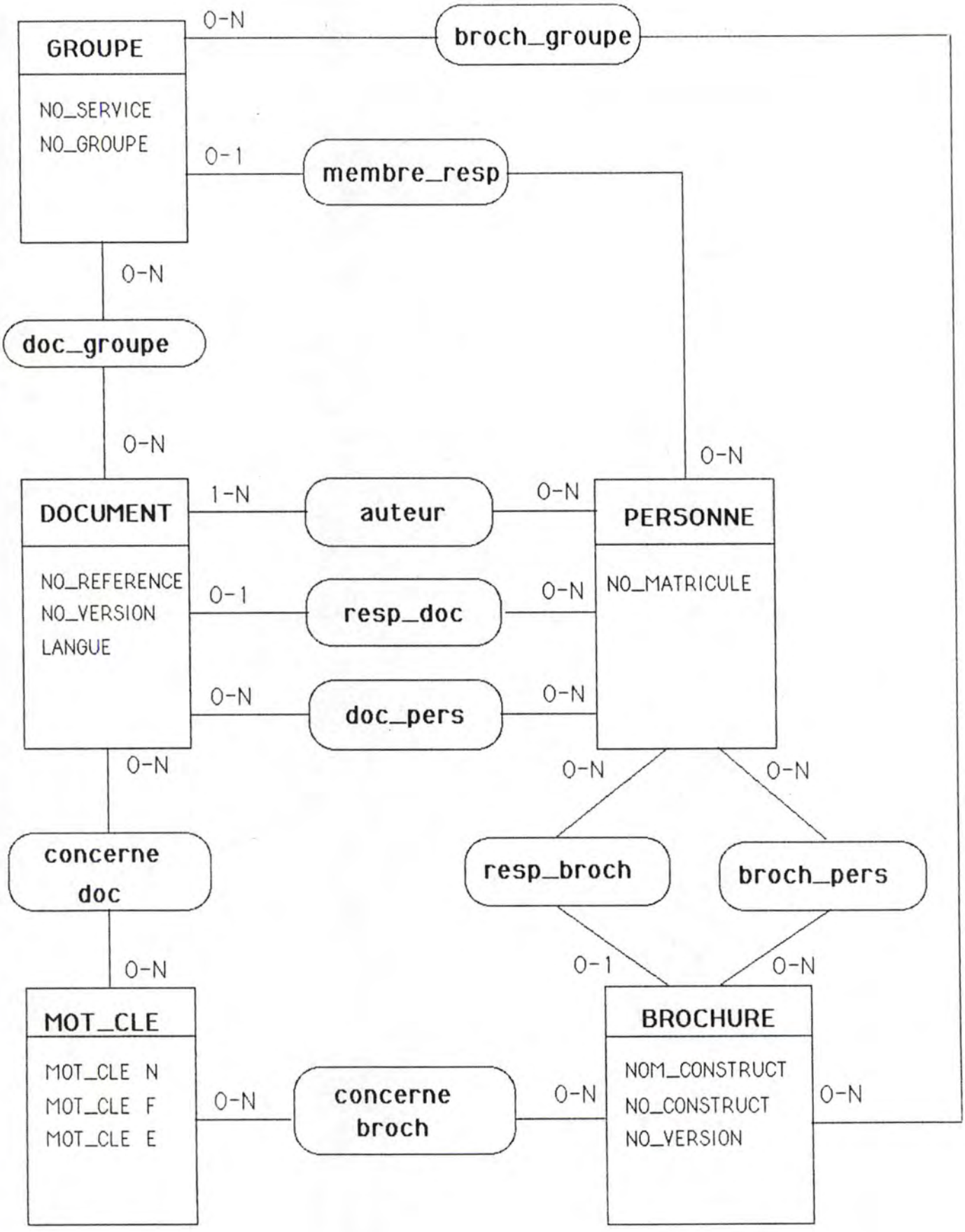
Par ailleurs, certains membres du M.A.I. ont collaboré à la réalisation du schéma conceptuel des données et à la complétude des spécifications des traitements.

5.1. ANALYSE CONCEPTUELLE DES DONNEES

Nous allons tout d'abord présenter le schéma conceptuel des données résultant des réunions avec les utilisateurs. Ensuite, nous donnerons la définition des entités et des relations ainsi que des attributs qui les composent.

5.1.1. SCHEMA CONCEPTUEL DES DONNEES

Sur ce schéma, nous n'avons retenu que les attributs identifiants des entités et nous avons omis volontairement les attributs éventuels des relations. Nous avons repris ces attributs au niveau de la définition des entités et des relations. A la suite du schéma, nous avons établi la liste des contraintes d'intégrité les plus importantes pour la cohérence du système et qui ne sont pas reprises sur le schéma contrairement aux identifiants et aux connectivités.


GROUPE
NO_SERVICE
NO_GROUPE
broch_groupe
membre_resp
doc_groupe
DOCUMENT
NO_REFERENCE
NO_VERSION
LANGUE
auteur
PERSONNE
NO_MATRICULE
resp_doc
doc_pers
concerne doc
resp_broch
broch_pers
MOT_CLE
MOT_CLE N
MOT_CLE F
MOT_CLE E
concerne broch
BROCHURE
NOM_CONSTRUCT
NO_CONSTRUCT
NO_VERSION
0-N
0-1
1-N

Les contraintes d'intégrité non reprises sur le schéma sont :

a) La relation DOC-PERS reliant les entités DOCUMENT et PERSONNE n'est pas identifiée par les identifiants des entités sur lesquelles elle est définie. Notons que cela constitue une dérogation à la sémantique du schéma telle qu'elle est proposée à la CGER. La sémantique a été décrite dans la première partie au chapitre 1. De même pour les relations suivantes : BROCH-PERS reliant les entités BROCHURE et PERSONNE, DOC-GROUPE reliant les entités DOCUMENT et GROUPE et BROCH-GROUPE reliant les entités BROCHURE et GROUPE.

b) Le document possède entre autres les attributs "type", "état" et "phase" du document. L'attribut "type du document" indique si le document est un original ou la traduction d'un document original, l'attribut "état" indique si le document est manuscrit (M) ou dactylographié (D) et l'attribut "phase" indique si le document est en rédaction (ER), rédigé (R), en traduction (ET), traduit (T), en dactylographie (ED), dactylographié (D), à la reproduction (ARP), reproduit (RP), distribué partiellement (DP) ou distribué totalement (DT). Nous avons prévu pour l'état et la phase du document une clause sans objet (SO) dans le cas où l'utilisateur aurait besoin d'un état ou d'une phase qui n'est pas prévue. Nous reprenons,ci-dessous, dans deux tables à double entrée (une par type de document) les couples autorisés (noté "X") et les couples interdits (notés "O").

Document ORIGINAL :

	ER	R	ET	T	ED	D	ARP	RP	DP	DT	SO
M	O	X	X	X	X	O	O	O	O	O	X
D	O	X	X	X	O	X	X	X	X	X	O
SO	X	O	O	O	O	O	O	O	O	O	X

Document TRADUCTION :

	ER	R	ET	T	ED	D	ARP	RP	DP	DT	SO
M	O	O	O	O	O	O	O	O	O	O	X
D	O	O	O	O	O	X	X	X	X	X	O
SO	O	O	O	O	X	O	O	O	O	O	X

c) le document possède aussi comme attribut le "nom de la personne qui modifie" actuellement le document courant. Il ne peut exister de modificateur pour un document original ou sa traduction de numéro de version "i" s'il existe une version originale dont le numéro de version est supérieur à "i" ou si la phase du document courant est "en rédaction".

5.1.2. DEFINITION DES ENTITES

a) Document

Un document est une information spécifique, présentée sous forme écrite, graphique ou mixte en français ou en néerlandais. Les documents demandent à être produits, dactylographiés, traduits et sont généralement classés, leur distribution se fait par des moyens asynchrones (poste interne ou externe).

Nous considérons un document comme étant une version d'un document dans une langue particulière.

Ses attributs sont :
- no-référence : numéro de référence attribué à un document de telle sorte que toutes les versions d'un document aient le même numéro de référence.
- no-version : numéro de version attribué à un document pour pouvoir distinguer les différentes versions d'un document.
- langue : langue dans laquelle le document est écrit afin de distinguer la version française de sa version néerlandaise (chaque document devant exister dans les deux langues nationales).
- titre du document
- type du document pour indiquer si c'est la version originale ou bien sa traduction.
- statut du document pour distinguer un document complet des modifications ne faisant pas l'objet d'un remaniement complet du document (addendum).
- phase du document pour permettre de suivre le cycle de vie du document. Le document peut être en rédaction, rédigé, en traduction, traduit, en dactylographie, dactylographié, en reproduction, reproduit, sans objet (dans tous les autres cas).
- état du document pour indiquer si le document est manuscrit, dactylographié, ou sans objet (surtout dans le cas du document en phase de rédaction pour un document original et en phase de traduction pour la traduction d'un document original).
- date d'entrée en phase.
- support du document pour localiser le document (références techniques) si ce dernier n'est pas sur papier.
- nom de la personne qui modifie pour les documents qui sont en cours de modification (facultatif).

b) Brochure

Une brochure est une information spécifique présentée sous forme écrite, graphique ou mixte en français, néerlandais, anglais ou allemand. Les brochures sont produites par les constructeurs et font l'objet d'une commande pour les obtenir. Elles sont généralement classées et leur distribution se fait par des moyens asynchrones (poste interne ou externe).

Nous considérons une brochure comme étant une version d'une brochure.

Ses attributs sont :
- nom-constructeur : nom du constructeur responsable de l'édition de la brochure.
- no-constructeur : numéro de référence de la brochure attribué par le constructeur.
- no-version : numéro de version de la brochure attribué par le constructeur.
- produit concerné par la brochure (DB2, APE,...).
- titre de la brochure
- release-number : numéro attribué par le constructeur pour indiquer la version du produit concerné par la brochure.
- rôle linguistique : langue dans laquelle la brochure est rédigée (français, néerlandais, anglais ou allemand).

c) Personne

Toute personne physique employée à la CGER.

Ses attributs sont :
- no-matricule : numéro identifiant toute personne au sein de la CGER.
- nom de la personne.
- prénom de la personne.
- no-service : numéro identifiant, au sein de la CGER, le service dans lequel la personne est employée.
- rôle linguistique de la personne.

d) Groupe

Un groupe est un ensemble de personnes qui sont réunies pour réaliser le même objectif. Une même personne peut appartenir à plusieurs groupes simultanément.
Exemples de groupe : un service, le C.T.I., le D.S.I., un groupe de travail, ...

Ses attributs sont :
- no-service : soit un numéro identifiant, au sein de la CGER, le service dans lequel l'ensemble des personnes du groupe sont employées, soit 0 si les membres du groupe sont éparpillés dans différents services.
- no-groupe : soit un numéro identifiant le groupe au sein d'un service, soit 0 si le groupe correspond au service lui-même, soit un numéro identifiant le groupe au sein de la CGER si les membres du groupe sont éparpillés dans différents services.
- dénomination française du groupe.
- dénomination néerlandaise du groupe.

e) Mot-clé

Un mot-clé est un mot significatif du sujet traité dans un document ou une brochure et peut être en français, néerlandais, anglais.

Ses attributs sont :
- mot-clé français
- mot-clé néerlandais
- mot-clé anglais

5.1.3. DEFINITION DES RELATIONS

a) Convention

R (E1,E2,...,En) pour désigner une relation définie sur les entités E1, E2, ..., En non nécessairement distinctes.

b) DOC-GROUPE (DOCUMENT,GROUPE)

On a une occurrence de la relation entre un document donné et un groupe donné soit si le document est distribué (dans le cas où il a déjà été envoyé) soit si le document est destiné (dans le cas où il n'a pas encore été envoyé) au groupe.

Ses attributs sont :
- nombre d'exemplaires destinés ou distribués.
- date d'envoi des exemplaires lorsqu'ils ont déjà été distribués ou 0 sinon.

c) DOC-PERS (DOCUMENT,PERSONNE)

Même chose que DOC-GROUPE (DOCUMENT,GROUPE) en remplaçant groupe par personne.

Remarque : Avec ces deux relations, on obtient la liste de distribution complète d'un document. Cette liste est donc constituée soit de groupes, soit de personnes, soit des deux.

d) BROCH-GROUPE (BROCHURE,GROUPE)

Même chose que DOC-GROUPE (DOCUMENT,GROUPE) en remplaçant document par brochure.

e) BROCH-PERS (BROCHURE,PERSONNE)

Même chose que DOC-PERS (DOCUMENT,PERSONNE) en remplaçant document par brochure.

Même remarque que précédemment mais pour la liste de distribution des brochures.

f) AUTEUR (DOCUMENT,PERSONNE)

On a une occurrence de la relation entre un document donné et une personne donnée si la personne est auteur de ce document.

Elle a pour attribut, la date de participation au document.

g) RESP-DOC (DOCUMENT,PERSONNE)

On a une occurrence de la relation entre un document donné et une personne donnée si la personne est responsable de ce document.

Elle a pour attribut, la date de prise de responsabilité.

h) RESP-BROCH (BROCHURE,PERSONNE)

Même chose que RESP-DOC (DOCUMENT,PERSONNE) en remplaçant document par brochure.

h) MEMBRE-RESP (GROUPE,PERSONNE)

Même chose que RESP-DOC (DOCUMENT,PERSONNE) en remplaçant document par groupe.

i) CONCERNE-DOC (DOCUMENT,MOT-CLE)

On a une occurrence de la relation entre un document donné et un mot-clé donné si le mot-clé est significatif du sujet traité dans le document.

j) CONCERNE-DOC (DOCUMENT,MOT-CLE)

Même chose que CONCERNE-DOC (DOCUMENT,MOT-CLE) en remplaçant document par brochure.

5.2. ANALYSE CONCEPTUELLE DES TRAITEMENTS

Dans un premier temps, nous avons dégagé deux gestions principales, l'une concernant les documents, l'autre concernant les brochures, pour permettre à l'utilisateur de manipuler les informations y afférentes (les attributs, d'une part, les personnes, les groupes et les mots-clés en relation avec les documents et brochures, d'autre part). Pour que l'utilisateur puisse manipuler des personnes, des groupes et des mots-clés autrement que par l'intermédiaire des documents et des brochures, nous avons ajouté par la suite la gestion de chacun de ces objets.

De ce fait, on retrouve un type de gestion par entité où les fonctions associées à la gestion des documents et des brochures ont une portée plus puissante que les fonctions associées à la gestion des autres entités. Chacune de ces gestions corresponde à une "activité" en regard au but permanent et essentiel qu'elle poursuive dans le système : gérer. Nous invitons le lecteur à se référer au chapitre 1 de la première partie, au point concernant l'étude de l'existant pour les définitions d'"activité" et de "tâche".

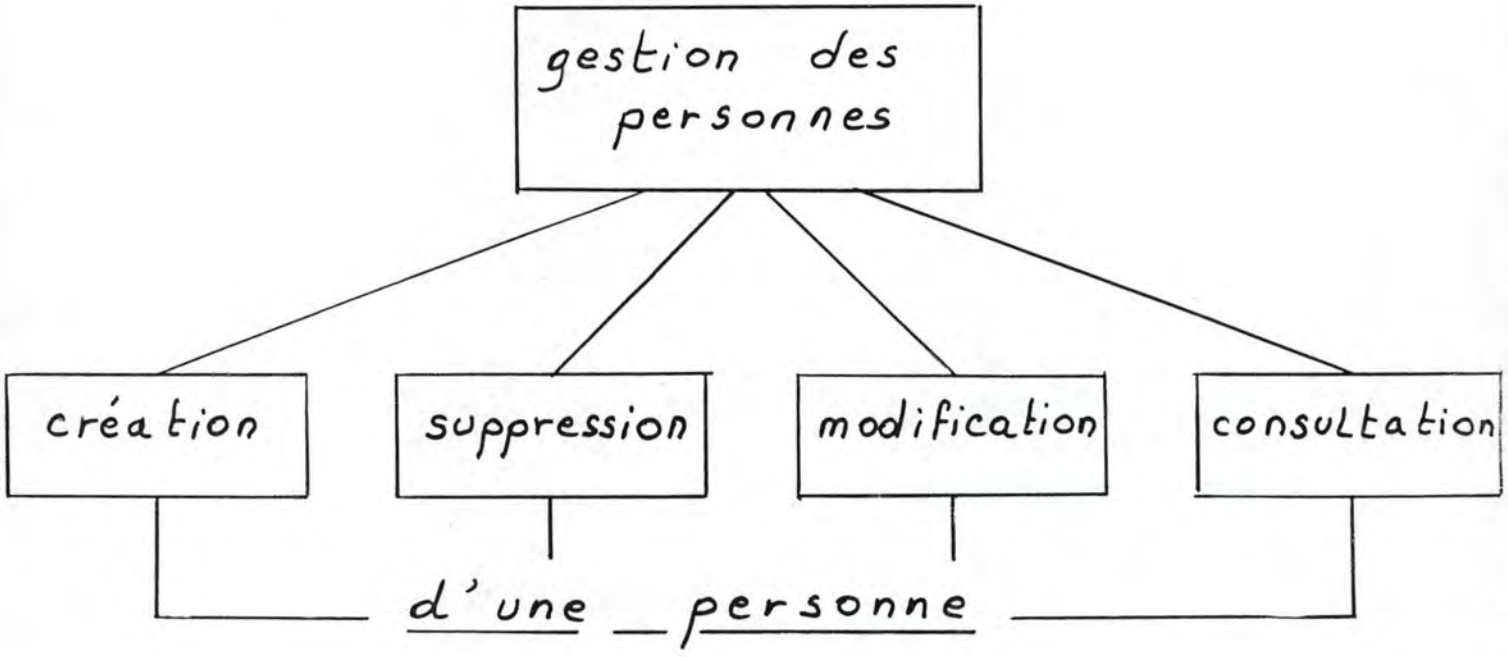
Ensuite, chaque type de gestion a été découpé en sous-traitements relatifs à des fonctions de base : création, suppression, modification et consultation de l'objet sur lequel porte la gestion. La suppression, la modification et la consultation correspondent à des "tâches" car elle s'exécutent sans interruption (unicité de temps), au même poste de travail (unicité de lieu) et en suivant des règles prédéfinies (unicité d'action). Ce dernier critère n'est pas respecté dans le cas de la création.

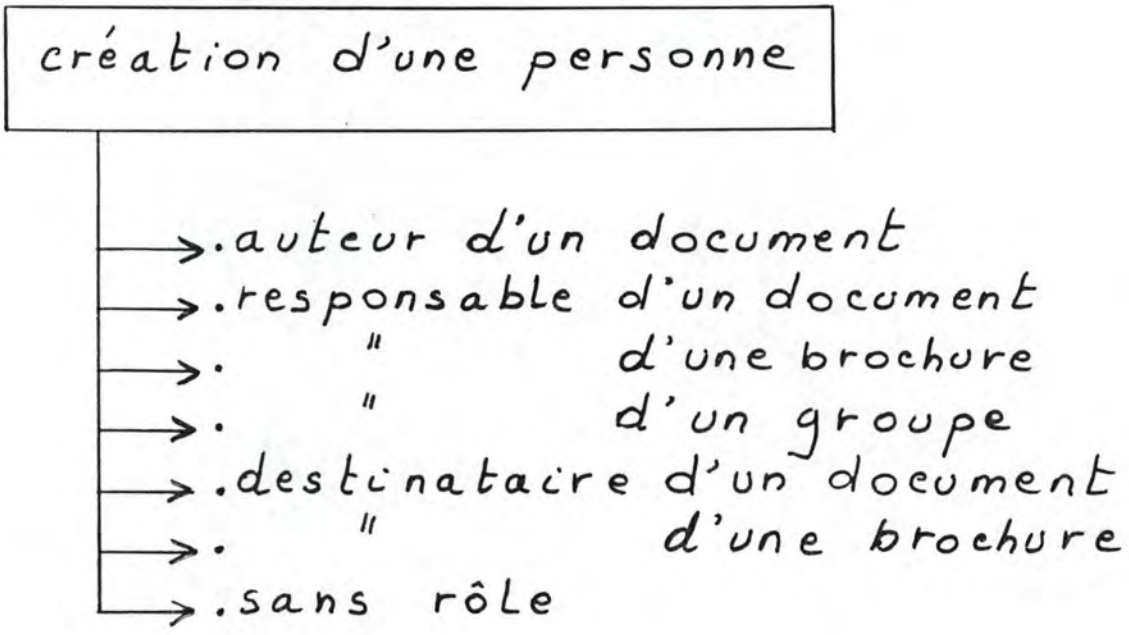
La spécification de ces quatre fonctions de base au sein de la "gestion d'une personne" vont nous permettre de confirmer ces allégations. Cependant, le lecteur intéressé par la spécification complète des cinq types de gestion pourra consulter l'annexe D.

Nous reprenons ci-dessous le schéma de la découpe de l'activité "gestion d'une personne", qui constitue le sous-système que nous avons implémenté par la suite (cfr. infra). Ensuite, nous donnerons la spécification des ses composants ainsi que la spécification de quatre traitements que ces composants utilisent.

La méthodologie proposait le schéma HIPO pour établir les spécifications mais celui-ci n'étant pas approprié à notre type de problème, nous avons préféré procéder autrement en essayant de donner une vue claire des fonctionnalités du système.

5.2.1. DECOUPE DE L'ACTIVITE "GESTION D'UNE PERSONNE"





5.2.2. EXISTENCE D'UN DOCUMENT (no-référence,no-version,langue)

Cette fonction permet de savoir si un document de no-référence, no-version, langue donnés existe ou pas dans le système.

5.2.3. EXISTENCE D'UNE BROCHURE (nom-constructeur, no-constructeur,no-version)

Cette fonction permet de savoir si une brochure de nom-constructeur, no-constructeur, no-version donnés existe ou pas dans le système.

5.2.4. EXISTENCE D'UN GROUPE (no-service,no-groupe)

Cette fonction permet de savoir si un groupe de no-service, no-groupe donnés existe ou pas dans le système.

5.2.5. IDENTIFICATION D'UNE PERSONNE (no-matricule)

Cette fonction permet de savoir si une personne de no-matricule donné existe ou pas dans le système. Si elle n'existe pas, le système crée une nouvelle occurrence de personne identifiée par le no-matricule donné et l'utilisateur devra compléter la description de la personne avec son nom, son prénom, son no-service et son rôle linguistique.

5.2.6. CREATION D'UNE PERSONNE

Une personne peut être créée en lui associant un ou plusieurs rôles ou simplement sans rôle du tout.

L'utilisateur doit fournir les informations suivantes :
- no-matricule,
- rôles possibles que jouera la personne dans le système :
  1. auteur d'un document
  2. responsable d'un document
  3. responsable d'une brochure
  4. responsable d'un groupe
  5. destinataire d'un document
  6. destinataire d'une brochure
  7. personne sans rôle

et
- pour les rôles 1, 2 et 5 : no-référence, no-version et langue
- pour les rôles 3 et 6 : nom-constructeur, no-constructeur et no-version.
- pour le rôle 4 : no-service et no-groupe.

En fonction du rôle que l'utilisateur aura assigné à la personne, le processus de création sera différent et c'est la création suivant le rôle qui correspond à une "tâche" puisqu'à ce moment, on retrouve l'unicité d'action. Pour les décrire, nous les classerons en 4 catégories: auteur, responsable, destinataire, sans rôle.

a) Auteur d'un document

```
Existence d'un document (no-référence,no-version,langue)
Si le document n'existe pas
   alors la création est impossible
   sinon identification d'une personne (no-matricule)
         Si la personne existe et est déjà auteur du
            document
            alors la création est inutile
            sinon création de la relation auteur avec la
                  date de participation fournie par
                  l'utilisateur.
```

b) Responsable d'un document

```
Existence d'un document (no-référence,no-version,langue)
Si le document n'existe pas
   alors la création est impossible
   sinon si le document a déjà un responsable
            alors la création est impossible
            sinon identification d'une personne (no-
                  matricule) et création de la relation
                  responsable avec la date de prise de
                  responsabilité fournie par l'utili-
                  sateur.
```

Même chose pour responsable d'une brochure ou d'un groupe en remplaçant document respectivement par brochure et groupe et leurs identifiants.

c) Destinataire d'une brochure

```
Existence d'un document (no-référence,no-version,langue)
Si le document n'existe pas
   alors la création est impossible
   sinon identification d'une personne (no-matricule) et
         création de la relation destinataire avec le
         nombre d'exemplaires et la date d'envoi fournis
         par l'utilisateur.
```

Même chose pour destinataire d'une brochure en remplaçant document par brochure et leurs identifiants.

5.2.7. SUPPRESSION D'UNE PERSONNE

L'utilisateur doit fournir le no-matricule de la personne. Si la personne n'existe pas dans le système alors la suppression est impossible sinon suppression de la personne et de ses éventuels rôles qu'elle a dans le système sauf si elle se trouve être l'unique auteur d'un document car un document doit avoir au moins un auteur (cfr. schéma conceptuel des données, page 131).

5.2.8. MODIFICATION D'UNE PERSONNE

L'utilisateur doit fournir le no-matricule de la personne. Si la personne n'existe pas alors la modification est impossible sinon l'utilisateur pourra modifier les informations existantes pour cette personne (nom, prénom, no-service, rôle linguistique).

5.2.9. CONSULTATION D'UNE PERSONNE

L'utilisateur doit fournir le no-matricule de la personne. Si la personne n'existe pas alors la consultation est impossible sinon elle fournit les informations la concernant (nom, prénom, no-service, rôle linguistique) et les rôles auxquelles elle participe (auteur, responsable d'un document, responsable d'une brochure, responsable d'un groupe, destinataire d'un document, destinataire d'une brochure ou tout simplement sans rôle).

A présent que nous avons décrit l'analyse conceptuelle des données et des traitements, nous pouvons aborder la phase suivante : l'analyse fonctionnelle.

6. L'ANALYSE FONCTIONNELLE

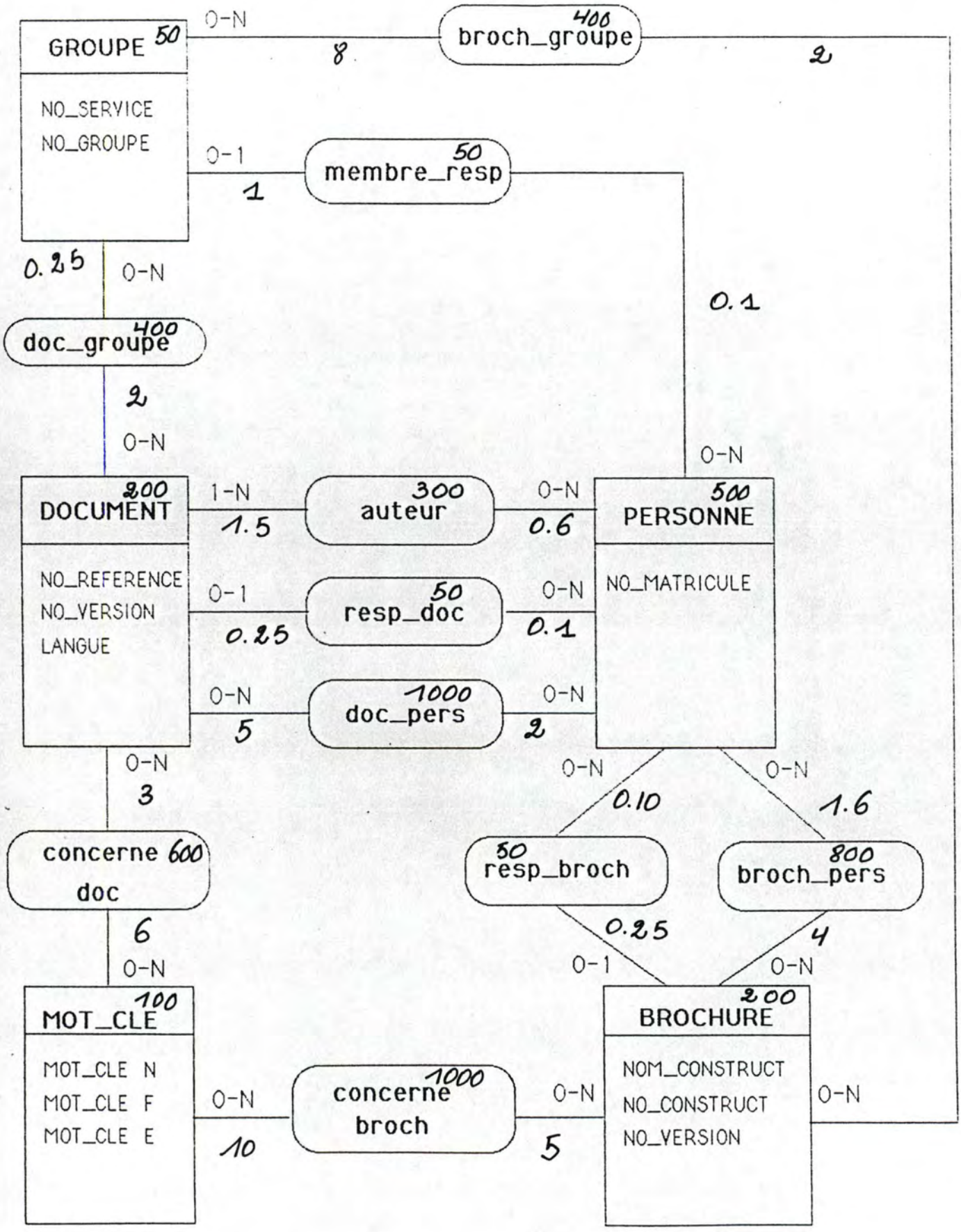
Nous avons réalisé pour cette étape, les quantifications statiques des données, le design des menus et des écrans et le dialogue utilisateur-système associé à chaque écran.

6.1. LES QUANTIFICATIONS STATIQUES DES DONNEES

Ces quantifications sont basées sur quelques indications qui nous ont été fournies par l'utilisateur. Ces indications concernent surtout les documents, les brochures, les auteurs et les responsables. Pour les éléments au sujet desquels nous n'avions aucun renseignement, nous avons tenté d'avancer des chiffres sur base de la logique du problème.

Le modèle conceptuel quantifié (volumes des types d'entités, des types de relations, pourcentage du nombre de T.A. par T.E.) représenté à la fig. 1. est le résultat de toute une démarche d'estimation de ces volumes décrite en 6.1.1. et 6.1.2.

Nous avons mené notre analyse en deux temps. La première phase s'attache aux documents, aux brochures et à leurs associations, pour lesquels nous disposions de plus amples informations. La seconde est relative aux personnes, aux groupes et aux mots-clés; pour ces T.E., nous décrirons des quantifications sur base des différents volumes dégagés dans la première phase.


GROUPE 50
NO_SERVICE
NO_GROUPE
0-N
8
broch_groupe 400
2
0-1
1
membre_resp 50
0.25
0-N
0.1
doc_groupe 400
2
0-N
0-N
DOCUMENT 200
NO_REFERENCE
NO_VERSION
LANGUE
1-N
1.5
auteur 300
0-N
0.6
PERSONNE 500
NO_MATRICULE
0-1
0.25
resp_doc 50
0-N
0.1
0-N
5
doc_pers 1000
0-N
2
0-N
3
0-N
0.10
0-N
1.6
concerne 600
doc
resp_broch 50
broch_pers 800
6
0.25
4
0-N
0-1
0-N
MOT_CLE 100
MOT_CLE N
MOT_CLE F
MOT_CLE E
0-N
10
concerne 1000
broch
0-N
5
BROCHURE 200
NOM_CONSTRUCT
NO_CONSTRUCT
NO_VERSION
0-N

6.1.1. LES DOCUMENTS ET LES BROCHURES

a) Les documents

Pour ce type d'entité, le système renferme 200 documents ou plus exactement, 200 versions de documents.

Pour chaque type d'association attaché au type d'entité document, nous reprendrons un mode de représentation particulier que nos traduirons, à titre d'exemple, pour le T.A. auteur.

Pour le T.A. auteur :
- 300 A. Auteur (moyenne 1.5 auteur par document)
- 30 auteurs (membres du service M.A.I.)

Traduction : sachant qu'il y a 200 documents et à raison d'une moyenne de 1.5 auteur par document, nous déduisons qu'il y a 300 associations de type auteur. Par ailleurs, un auteur est une personne reprise parmi les 30 membres du M.A.I.; ceci nous permettra par la suite d'évaluer le nombre total de personnes reprises dans le système.

Ainsi, nous pouvons définir les quantifications pour les autres T.A.

Pour le T.A. resp-doc :
- 50 A. resp-doc (responsable mentionné pour 1 document sur 4)
- 30 responsables (membres du service M.A.I.)

Pour le T.A. doc-pers :
- 1000 A. doc-pers (5 destinataires par document)
- 300 personnes destinataires

Pour le T.A. doc-groupe :
- 400 A. doc-groupe (2 groupes destinataires par document)
- 50 groupes destinataires

Pour le T.A. mot-clé :
- 600 A. mot-clé (3 mots-clés par document)
- 50 mots-clés

b) Les brochures

Le système renferme 200 brochures (les différentes versions comprises).

Pour le T.A. resp-broch :
- 50 A. resp-broch (responsable mentionné pour 1 brochure sur 4)
- 30 responsables (membres du services M.A.I.)

Pour le T.A. broch-pers :
- 800 A. broch-pers (4 destinataires par brochure)
- 200 personnes destinataires

Pour le T.A. broch-groupe :
- 400 A. broch-groupe (2 groupes destinataires par brochures)
- 50 groupes destinataires

Pour le T.A. mot-clé :
- 1000 A. mot-clé (5 mots-clés par brochure)
- 80 mots-clés

6.1.2. LES PERSONNES, GROUPES ET MOTS-CLES

Nous avons déduit les volumes pour chacun de ces T.E. sur base des indications fournies par la quantification des documents et des brochures. Pour ce, nous avons utilisé la fonction d'union (U) dont le résultat est le fruit d'approximations qui se basent sur la logique du système.

a) Les mots-clés

(50 mots-clés pour les documents) U (80 mots-clés pour les brochures) = 100 mots-clés

b) Les groupes

(50 groupes pour les documents) U (50 groupes pour les brochures) = 50 groupes

c) Les personnes

(30 membres du service M.A.I.) U (40 responsables de groupes) U (300 destinataires de documents) U (200 destinataires de brochures) = 500 personnes

Il est utile de garder la démarche dans le cas où l'on modifierait une des hypothèses du système (par exemple, ne plus considérer les seuls membres du M.A.I. comme auteurs mais l'ensemble des membre du DSI). Par ailleurs, nous n'aurions jamais pû trouver, par exemple, que 10 % des personnes étaient responsables d'une brochure sans une analyse plus fine.

Nous n'avons pas réalisé de quantifications dynamiques car il était quasi impossible d'évaluer la fréquence journalière pour chacun des traitements. Dans le cas où il est possible de faire des estimations de manière plus précise, ces quantifications sont alors utilisées pour la phase de design physique de la base de donnée. En effet, des mécanismes d'accès performant pourront être instaurés pour les tâches les plus fréquentes et les plus coûteuses.

6.2. DESIGN DES MENUS ET DES ECRANS

A ce niveau, nous avons utilisé un outil de développement d'applications interactives APE (Application Prototype Environment) qui a été décrit de manière succincte au chapitre 2 de la première partie.

Cet outil se base notamment sur la notion d'enchaînement de menus et d'écrans auxquels on associe des fonctions écrites en APL.

Ce logiciel a été choisi parce qu'il est sensé accélérer le développement des applications et parce qu'il est simple à utiliser (pour les utilisateur qui ont des notions d'APL). Par ailleurs, il permet de réaliser facilement des écrans conviviaux (couleurs,...).

Pour la phase de l'analyse fonctionnelle des traitements, nous nous sommes limités à utiliser APE comme un "traitement de texte" en vue de générer les menus et les écrans nécessaires à l'ensemble de l'application; ces écrans constituant le support au dialogue utilisateur-système.

Nous verrons, par la suite (analyse technique), comment ces différents menus et écrans seront utilisés afin de fournir à l'utilisateur les fonctions qu'il a spécifiées.

6.3. LE DIALOGUE UTILISATEUR-SYSTEME

Nous avons réalisé le dialogue utilisateur-système sur base des spécifications conceptuelles et des écrans.

La méthodologie ne propose pas de démarche pour réaliser cette étape, aussi nous avons adopté une démarche orientée utilisateur c'est-à-dire où l'on explique à l'utilisateur comment il doit se comporter face au système. Pour ce, nous avons décrit globalement les opérations effectuées par le système lui-même. La manière dont celles-ci sont réalisées reste donc transparente pour l'utilisateur.

Le lecteur intéressé par la spécification complète du dialogue utilisateur-système associé à l'ensemble de l'application pourra consulter l'annexe F. Nous reprenons cependant, ci-dessous, les dialogues utilisateur-système associés à la création d'une personne auteur d'un document à partir des écrans INTRODOC, INTRODOCPERS, AUTEUR1 et AUTEUR2 et les différents menus nécessaires. Ceci en vue de donner un exemple complet de dialogue vis-à-vis d'une tâche.

```
  MENU (MENU1) 86-12-23 11.10 -   -
================================================================================
PUMA                                GESTION DES DOCUMENTS
MENU1                               ======================


QUEL TYPE D'OPERATION DESIREZ-VOUS EFFECTUER ?

 1 CREATION D'UN DOCUMENT
 2 SUPPRESSION D'UN DOCUMENT
 3 MODIFICATION D'UN DOCUMENT
 4 CONSULTATION D'UN DOCUMENT
 X TERMINAISON

                      SELECTION :

PF: 2=COMMANDS 3=END
--------------------------------------------------------------------------------
```

```
  MENU (MENU31) 86-12-29 09.25 -   -
================================================================================
PUMA                                CREATION D'UN GROUPE
MENU31                              ====================


QUEL ROLE DESIREZ-VOUS ASSIGNER AU GROUPE A CREER ?

 1 DESTINATAIRE D'UN DOCUMENT
 2 DESTINATAIRE D'UNE BROCHURE
 3 SANS ROLE
 X TERMINAISON

                      SELECTION :

PF: 2=COMMANDS 3=END
--------------------------------------------------------------------------------
```

```
  MENU (MENU51) 86-12-29 14.33 -   -
================================================================================
PUMA                                CREATION D'UN MOT-CLE
MENU51                              =====================


QUEL ROLE DESIREZ-VOUS ASSIGNER AU MOT-CLE A CREER ?

 1 CONCERNE UN DOCUMENT
 2 CONCERNE UNE BROCHURE
 3 SANS ROLE
 X TERMINAISON

                      SELECTION :

PF: 2=COMMANDS 3=END
--------------------------------------------------------------------------------
```

```
MENU <MENU41> 86-12-29 10.55 -  -
================================================================================

PUMA                              CREATION D'UNE PERSONNE
MENU41                            ========================

QUEL ROLE DESIREZ-VOUS ASSIGNER A LA PERSONNE A CREER ?
 1 AUTEUR D'UN DOCUMENT
 2 RESPONSABLE D'UN DOCUMENT
 3 RESPONSABLE D'UNE BROCHURE
 4 RESPONSABLE D'UN GROUPE
 5 DESTINATAIRE D'UN DOCUMENT
 6 DESTINATAIRE D'UNE BROCHURE
 7 SANS ROLE
 X TERMINAISON

                        SELECTION :

PF: 2=COMMANDS 3=END
--------------------------------------------------------------------------------
```

```
    △_DOC 'F:INTRODOC'
Panel (INTRODOC) 87-05-04 18.06 - DEMANDE D'IDENTIFICATION D'UN DOCUMENT -
```

< titre du panel >
NO-REFERENCE : NO-VERSION : LANGUE :

Le système affiche le panel INTRODOC
avec Titre du panel =

1. SUPPRESSION D'UN DOCUMENT si l'utilisateur a choisi l'option 2
2. MODIFICATION " " " " " " " 3
3. CONSULTATION " " " " " " " 4.

dans MENU 1

4. GROUPE DESTINATAIRE D'UN DOCUMENT si l'utilisateur a choisi l'option 1
dans MENU 31
5. AUTEUR D'UN DOCUMENT si l'utilisateur a choisi l'option 1
6. RESPONSABLE D'UN DOCUMENT si l'utilisateur a choisi l'option 2
7. PERSONNE DESTINATAIRE D'UN DOCUMENT si l'utilisateur a choisi l'option 5

dans MENU 41

8. MOT_CLE CONCERNANT UN DOCUMENT si l'utilisateur a choisi l'option 1
dans MENU 51

1) L'utilisateur introduit le no-référence, no-version et langue du document et presse ENTER
2) Si les données sont correctement introduites, le système procède a un contrôle d'existence du document ainsi identifié.

Si # alors Msg : "ce document n'existe pas" et réaffichage d'INTRODOC

Si ∃ alors pour 1. SUPDOC
2. MODDOC
3. CSLTDOC
4. INTRO GRP DEST DOC
5. INTRO DOCPERS
6. INTRO DOC PERS
7. INTRO DOCPERS
8 INTRO MOT CONC DOC

⚠ pour 6. on affiche le panel INTRODOCPERS
si le document n'a pas encore de responsable
sinon réaffichage d'INTRODOC

```
△_DOC 'F:INTRODOCPERS'
Panel <INTRODOCPERS> 87-04-27 18.43 - INTRODUCTION PERSONNE LIEE A DOCUMENT -
```

< titre du panel >
NO-REFERENCE : NO-VERSION : LANGUE :
NO-MATRICULE :

Le système affiche le panel INTRODOCPERS
avec Titre du panel =

1. AUTEUR D'UN DOCUMENT
2. RESPONSABLE D'UN DOCUMENT
3. PERSONNE DESTINATAIRE D'UN DOCUMENT

selon qu'il suit le panel INTRODOC de même ~~nom~~ titre

1) le système génère les champs :
no-référence, no-version et langue

2) l'utilisateur introduit le no-matricule et presse ENTER.

3) si les données sont correctement introduites
le système procède au contrôle d'existence de la personne
ainsi identifiée

si ‖ # ds le système alors pour 1. AUTEUR1 ;
pour 2. RESPDOC1 ; pour 3. PSNDESTDOC1
‖ ∄ ds le système alors pour 1. AUTEUR2 ;
pour 2. RESPDOC2 ; pour 3. PSNDESTDOC2

```
     ▵_DOC 'F:AUTEUR1'
  Panel (AUTEUR1) 87-05-04 18.08 - AUTEUR D'UN DOCUMENT 1 -
```

```
+-----------------------------------------------------------------------------
!
!
!                                AUTEUR D'UN DOCUMENT
!                                ====================
!
!
!NO-REFERENCE :                    NO-VERSION :                  LANGUE :
!
!
!NO-MATRICULE :
!
!       NOM                  :
!       PRENOM               :
!       NO-SERVICE           :
!       ROLE LINGUISTIQUE    :
!
!
!DATE DE PARTICIPATION :
+-----------------------------------------------------------------------------
```

Le système affiche le panel AUTEUR1

1) le système génère les champs no-ref, no-vers, lg, nomat

2) l'utilisateur introduit le nom, prénom, no-service, date rôle linguistique et presse ENTER.

3) Si les données sont correctement introduites, le système créé la personne et la relation avec le document. Ensuite retour au MENU41

```
     △_DOC 'P:AUTEUR2'
  Panel <AUTEUR2> 87-05-04 11.03 - AUTEUR D'UN DOCUMENT 2 -
```

AUTEUR D'UN DOCUMENT

NO-REFERENCE : NO-VERSION : LANGUE :

NO-MATRICULE :

NOM :
PRENOM :
NO-SERVICE :
ROLE LINGUISTIQUE :

DATE DE PARTICIPATION :

Le système affiche le panel AUTEUR2

1) le système génère les champs:
no_référence, no_version, langue, no_matricule, nom, prénom, no_service, rôle linguistique

2) l'utilisateur introduit la date et presse ENTER

3) Si les données sont correctement introduites alors création de la relation avec le document si celle-ci n'existe pas déjà
Ensuite retour au MENU 41

7. L'ANALYSE TECHNIQUE

Pour cette phase, nous avons tout d'abord réalisé le schéma des données conforme au SGBD choisi à savoir DB2/SQL.

Ensuite, pour un sous-système de l'application à savoir l'activité "gestion d'une personne" nous avons établi l'enchaînement des menus et des écrans en répertoriant les accès à la base de données. Nous avons aussi relevé, sur base des écrans, les données qui devront être exploitées par une fonction APL. Nous avons alors décrit leur format leurs attributs (output, numeric input, literal input,... cfr. [IBM (a), p. 121) et les messages y associés en cas d'erreur syntaxique.

Pour terminer, nous avons décidé de la façon dont les écrans et les menus pourront être utilisés par l'utilisateur.

7.1. REALISATION DU SCHEMA DES DONNEES CONFORME A DB2/SQL

Cette phase consiste à traduire le schéma conceptuel des données en un schéma respectant les contraintes du SGBD choisi.

Les principales contraintes du SGBD DB2/SQL sont les suivantes :

- pas de chemins,
- pas de champ décomposable,
- pas de champ répétitif,
- au moins un champ par table.

En vue de faciliter cette transformation de schéma, nous avons tout d'abord traduit le schéma conceptuel du sous-système en un modèle d'accès généralisé (MAG) dont la sémantique est décrite dans [HAINAUT (b), p 20 28

Nous citerons ici les principaux concepts que nous avons utilisés pour réaliser ce modèle. Il s'agit des articles et types d'articles, d'items et valeurs d'item, des chemins d'accès et types de chemins d'accès et des identifiants composés. Etant donné que nous n'avons pas approfondi l'aspect performance, nous ne retiendrons pas ici la notion de clé d'accès.

La traduction du modèle conceptuel des données en MAG est représentée à la figure 1 (page 156). Notons que le MAG est limité aux types d'articles pour la clarté de la représentation. Cette dernière doit donc être consultée en regard de la liste des items pour chaque type d'article.

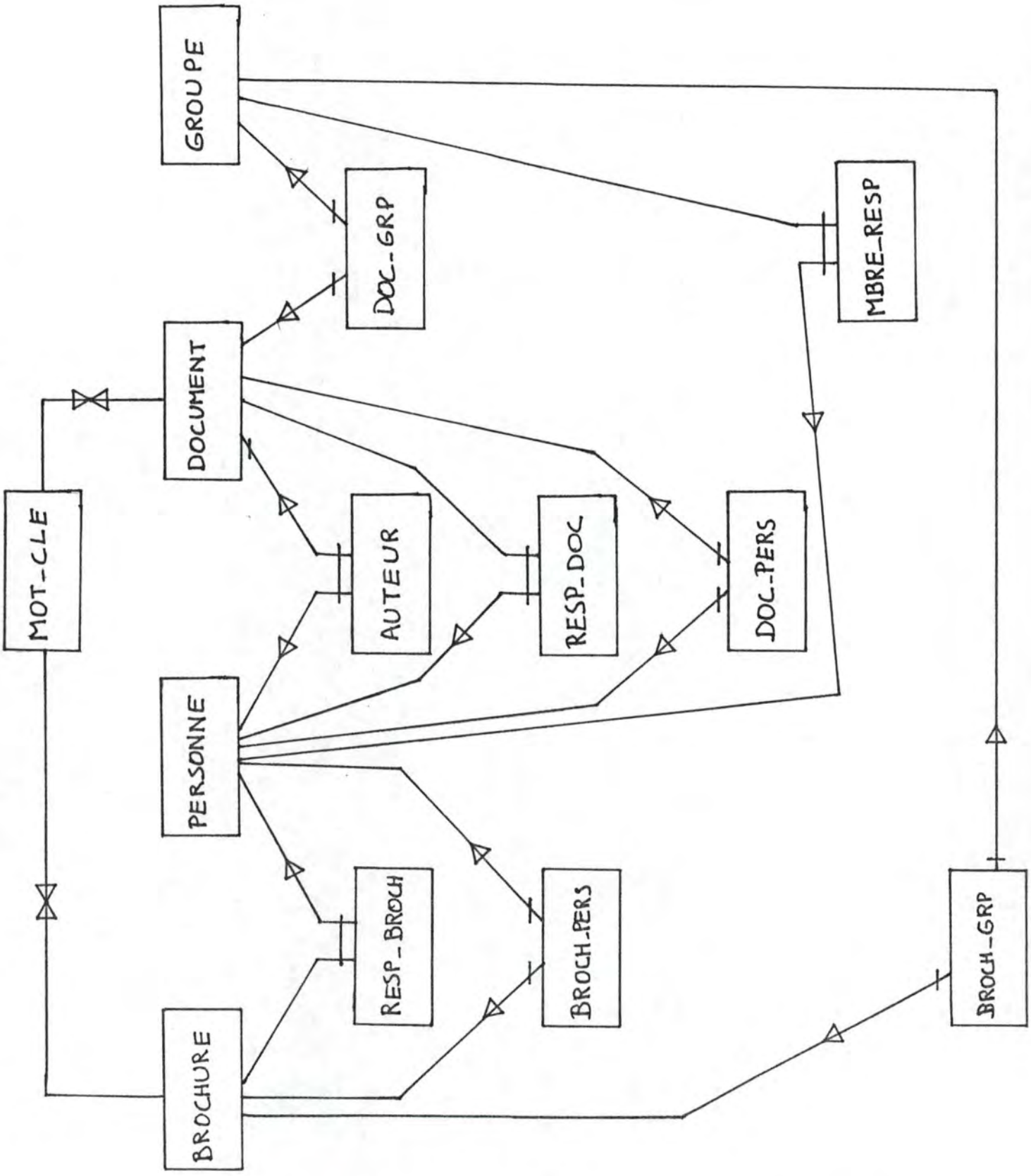


Fig. 1. : Modèle des accès généralisé (MAG)

Les conventions de représentation des concepts du MAG sont les suivantes :

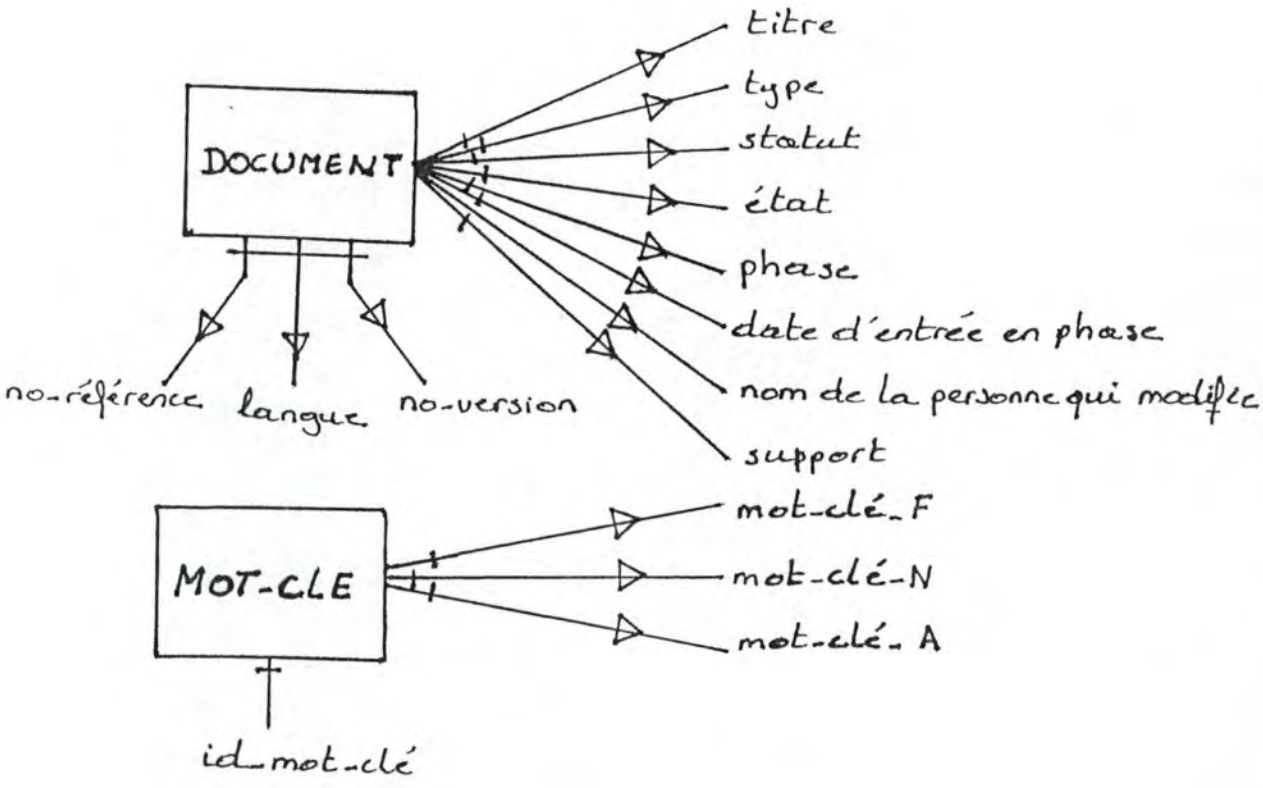
- un type d'article est représenté par une boîte,
- un item est représenté par son nom et relié à son type d'article par un arc
  - item obligatoire : barre sur l'arc
  - item répétitif identifiant : triangle
  - item simple non identifiant : triangle renversé
  - item répétitif non identifiant : diabolo
  - item simple identifiant : pas de symbole
  - items composés identifiants : parallélisme partiel des arcs
- un type de chemin est représenté par des arcs entre types d'articles, arcs caractérisés par leur classe fonctionnelle
  - 1-1 : pas de symbole
  - 1-N : triangle ( )
  - N-1 : triangle inverse ( )
  - N-N : diabolo

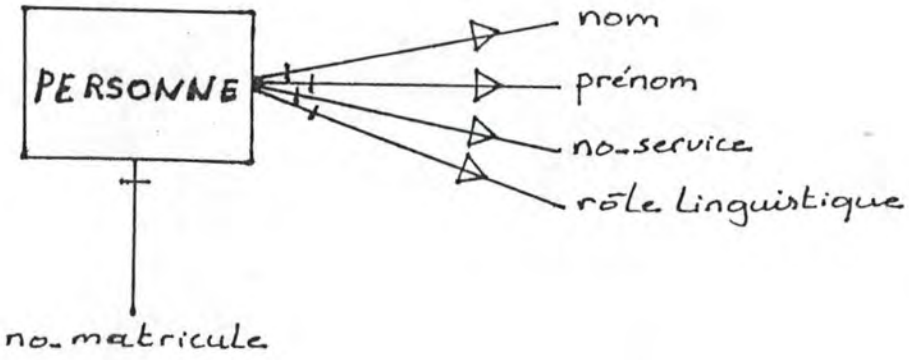
Pour réaliser le schéma conforme DB2/SQL à partir du MAG, nous avons supprimé tous les types de chemins; la figure 2 reprend quelques uns des types d'articles qui composent le schéma conforme DB2 (page 158). Les types de chemins de la classe fonctionnelle N-N ont dû être éclatés en nouveaux types d'articles (CONC-DOC et CONC-BROCH) dont les items identifiants sont constitués des identifiants des 2 types d'articles que chacun des types de chemins reliait (cfr. fig. 2, page 158). Dans le schéma conceptuel, nous avons considéré un identifiant groupé pour le T.E. mot-clé. Or pour ne pas charger les tables DB2, nous avons préféré créer un identifiant artificiel pour chaque groupe de valeurs. Cet identifiant sera inconnu de l'utilisateur.

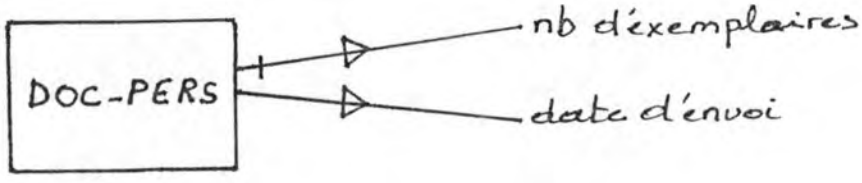
Nous n'avons pas étudié l'aspect performance, d'une part, parce que nous n'avions pas de quantifications de fréquences d'accès et, d'autre part, parce que DB2 offre la possibilité de créer et de supprimer des index à une table en cours d'exploitation de la BD. Ces index permettent en effet de diminuer le coût des accès; pour ce, il faut estimer l'utilité de placer un index sur base de plusieurs paramètres : fréquence d'accès, type d'accès, nombre de lignes et de colonnes de la table, longueur des champs,... Néanmoins, nous avons créé quelques index de type UNIQUE comme support d'identifiant dans tous les cas où le programme d'application n'assurait pas le contrôle de son unicité.

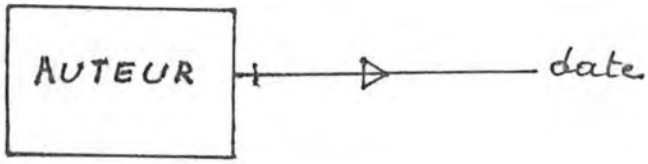
Le schéma conforme DB2/SQL doit être accompagné de la liste de contraintes destinées à conserver la sémantique du schéma conceptuel. Le respect de ces contraintes devra être assuré par le programme d'application ou, éventuellement par le SGBD (pour les identifiants).

Le lecteur intéressé par le schéma conforme à DB2/SQL accompagné de ses contraintes pourra consulter l'annexe F.







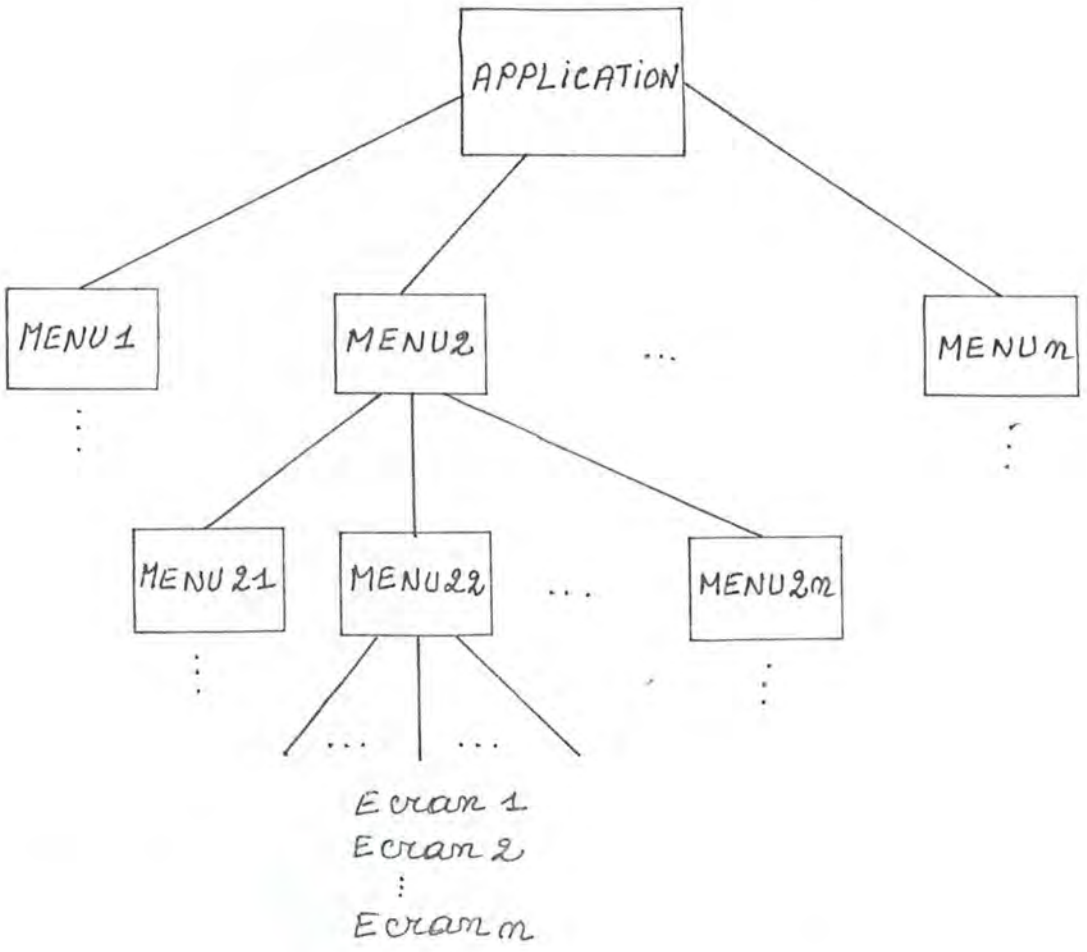


7.2. L'ENCHAINEMENT DES MENUS ET DES ECRANS

Nous ne reprendrons pas dans ce paragraphe le graphe des enchaînements des menus et des écrans décrit pour notre application car celui-ci ne serait pas compréhensible sans l'ensemble des menus et des écrans qui le composent. Aussi nous nous limiterons donc à présenter le principe. Cependant, le lecteur désireux de le consulter, le trouvera à l'annexe G1.

Nous pouvons considérer le graphe des enchaînements comme ayant trois niveaux : le niveau 1 correspondant à l'application PUMA et représenté par le menu principal, le niveau 2 représentant la découpe de l'application PUMA en activités et sous-activités et se matérialisant par des menus et le niveau 3 reprenant les écrans élaborés pour supporter les tâches.

Ce qui donne d'une manière schématique :



Au niveau 3, on retrouve donc les écrans qui ont servi de support au dialogue utilisateur-système. Nous avons donc pu à partir de ces informations, recenser les accès à la base de données ainsi que les données qui devront être exploitées par une fonction APL. Ces informations sont reprises à l'annexe G2.

7.3. ECRANS ET CHAMPS DE DONNEES

Nous avons établi alors, pour tous les écrans concernant le sous-système à implémenter, la liste des données qui devront être exploitées par une fonction APL. Cette liste comprend le nom de la donnée ("label"), son format, le message d'erreur a y associer pour le contrôle syntaxique. Cette liste figure à l'annexe G3.

7.4. UTILISATION DES MENUS ET DES ECRANS

7.4.1. UTILISATION DES MENUS

Pour déterminer facilement les opérations qu'il désire effectuer, l'utilisateur dispose d'une hiérarchie de menus dont l'enchaînement est géré par APE. Une opération consiste soit en l'exécution d'une fonction APL soit en l'affichage d'un nouveau menu.

On peut naviguer dans les menus de manière hiérarchique ou horizontale.

a) De manière hiérarchique

L'utilisateur mentionne dans la ligne de commande le numéro correspondant à une sélection dans le menu courant ou bien une série de numéros séparés par un point ou un blanc.

Exemple : 3.1.2. où
3 correspond à une sélection dans le menu courant
1 correspond à une sélection dans le menu suivant
2 correspond à une sélection dans le troisème menu

Si le signe = est mentionné au début, le processus commence à partir du menu du sommet dans le système.

b) De manière horizontale

L'utilisateur peut accéder directement à un menu en mentionnant dans la ligne de commande le nom de ce menu et ce à partir de n'importe quel endroit dans le système.

Avec la clé PF2 (touche fonction) , il peut demander la liste des noms de tous les menus et commandes qui peuvent être exécutés au niveau des menus. Ensuite, en positionnant le curseur à côté d'un nom de menu ou d'une commande et en pressant la clé PF6 (touche fonction), il peut accéder directement au menu désiré ou exécuter la commande.

L'utilisateur peut à tout moment utiliser la clé PF3 (touche fonction) pour accéder au menu précédant le menu courant dans la hiérarchie.

D'autre part, s'il fait appel à un menu inexistant ou s'il mentionne une sélection invalide dans un menu, le message suivant sera affiché dans le bas de son écran :

"INVALID SELECTION, COMMAND OR MENU CALL"

Pour poursuivre, l'utilisateur devra mentionner une sélection valide, appeler un menu existant ou exécuter une commane valide.

7.4.2. UTILISATION DES ECRANS

Dans cette application, toute opération sélectionnée via les menus entraîne l'affichage d'un ou plusieurs écrans. Un écran comprend des champs fixes et des champs variables. Parmi ces champs variables, certains sont vides et d'autres sont complétés par le système suivant l'opération associée à l'écran.

L'utilisateur peut compléter ou mettre à jour chacun des champs variables indiqué par le curseur. Il peut aussi faire véhiculer le curseur d'un champ à l'autre via " " ou " ".

Certaines zones sont définies comme zone "scroll" pour pouvoir faire défiler dans une fenêtre une liste de données. Les clés PF7 et PF8 (touches fonction) permettent respectivement de remonter ou de descendre dans la liste.

Quand l'écran est dûment complété, il faut presser ENTER, dès ce moment, une série de contrôles vont s'effectuer. Ainsi, un message explicatif apparaîtra au bas de l'écran si les informations introduites pour un champ ne sont pas syntaxiquement correctes ou si elles ne satisfont pas à certaines règles de conformité. De même, un message apparaîtra si un champ ou un groupe de champs identifiants ne vérifie pas certaines règles.

Ces messages doivent être prévus par le développeur. Si une de toutes ces règles n'est pas vérifiée, soit le curseur se positionne automatiquement sur le premier champ erroné, soit le système affiche l'écran adéquat selon le cas.

Si, à un moment donné, l'utilisateur désire quitter l'écran avant qu'il ne soit enregistré, il peut le faire à l'aide de la clé PF12 (touche fonction) qui aura pour effet de réafficher le menu sélectionné le plus récemment; toutes les opérations associées à l'écran courant ne s'exécuteront pas.

8. PROGRAMMATION ET TESTS

Cette étape consistait, d'une part, à implémenter la base de données à l'aide de QMF (voir les outils dans la première partie) et à tester les accès à la base de données ainsi construite et, d'autre part, à implémenter la succession des écrans en simulant les accès à la base de données.

On a ensuite intégrér les requêtes SQL dans les fonctions APL qui géraient la succession des écrans pour obtenir en fin de parcours un ensemble de fonctions associées au niveau 3 de l'enchaînement des menus et écrans. Cette étape s'est faite en parallèle avec les tests d'intégration.

Pour chaque fonction, nous avons finalement représenté sur un schéma les différents chemins possibles qu'elle contenait en vue, d'une part, de servir de documentation à la phase de programmation et, d'autre part, d'aider à la construction des jeux de test.

Le lecteur qui souhaite consulté l'ensemble du dossier relatif à la programmation et aux tests pourra consulter l'annexe H.

9. LES AMELIORATIONS

Dans ce point, nous décrirons quelques améliorations qui peuvent être apportées au système. Certaines d'entre elles ont été dégagées par les utilisateurs lors de la démonstration, d'autres avaient été envisagées en cours de développement mais nous les avions laissées de côté du fait de leur complexité d'implémentation.

Ci dessous, nous dressons la liste de ces améliorations :

- fonctions d'impression des résultats des consultations,
- fonction HELP pour documenter les champs à remplir,
- fonction de calcul (nombre total d'exemplaires à envoyer,...)
- accès sur base du nom de la personne, plutôt que sur no matricule,
- associer aux cinq gestions des fonctions d'opérations sur les relations,
- instaurer une fonction d'archivage (pour garder toutes les informations relatives à une ancienne version d'un document ou d'une brochure.

Notons qu'une application développée avec un logiciel APE peut aisément être étendue à faible coût s'il s'agit d'ajouter des fonctions élémentaires. Dans le cas de fonctions plus élaborées, comme l'archivage, un retour à l'analyse conceptuelle serait souhaitable.

CHAPITRE 2

LE DEVELOPPEMENT DU PROJET PUMA FACE A LA METHODOLOGIE PROPOSEE A LA CGER

1. INTRODUCTION

Dans ce chapitre, nous allons faire état des problèmes que nous avons rencontrés lors du développement de l'application PUMA du fait de l'utilisation de la méthodologie proposée par la CGER.

Dans ce but, nous avons passé en revue les différentes étapes de développement que nous avons suivies jusqu'à et y compris l'analyse fonctionnelle. Au-delà de cette étape, les problèmes survenus ne dépendaient plus de la méthodologie mais bien de l'application elle-même et des outils utilisés.

En fonction des problèmes dégagés, nous avons essayé d'apporter des améliorations à la méthodologie et non pas de procéder à un remodelage complet de la démarche proposée.

2. LA DEMANDE DE DEVELOPPEMENT

Nous avons éprouvé quelques difficultés à remplir ce formulaire en ce qui concerne l'établissement de quantifications et l'évaluation du budget (cfr. annexe D).

Les quantifications ont été estimées sur base de perspectives d'évolution du nombre de documents et de brochures. Mais à ce niveau, des quantifications ainsi déterminées sont suffisantes à l'évaluation de l'ampleur du projet. Sans celles-ci, la demande de développement perdrait sa raison d'être.

Le budget n'a été déterminé que pour les coûts en matière de personnel (sur base de standards hommes/mois). Les autres coûts étaient difficilement prévisibles à ce stade du développement. Nous pensons que le destinataire de la demande de développement considère toujours une marge d'erreur incluant les coûts additionnels. C'est lors du processus de choix que les décisions intermédiaires seront prises en vue de préserver le meilleur rapport coût-efficacité.

Par conséquent, il ne faut pas accorder un crédit démesuré à ces informations; ce document ne constitue qu'une ébauche à l'analyse du projet en vue d'obtenir l'approbation auprès de l'instance de décision.

Dans ce but, nous nous devons de conserver cette demande de développement dans la méthodologie. Nous pouvons d'ailleurs constater qu'elle est conçue de telle sorte que tout groupe de travail peut aisément l'adapter aux caractéristiques de son projet. De plus, les objectifs qui lui sont assignés doivent déterminer dans quelle mesure elle doit être complétée par les utilisateurs et les responsables du développement(cfr. annexe D).

3. LA DEFINITION DE PROJET

3.1. L'ETUDE DE L'EXISTANT

La CGER demande à ce niveau de recenser les activités, les principales tâches et les informations existantes et, ensuite, d'élaborer à partir des différents recensements une représentation schématique du système actuel pour servir de base à la critique. Mais avant d'aborder ces différentes phases, examinons la démarche préliminaire à notre étude de l'existant.

3.1.1. LA DEMARCHE

Pour réaliser cette étude, il était nécessaire de procéder à des interviews auprès des futurs utilisateurs. Aussi, la première difficulté que nous avons rencontrée était d'obtenir des personnes interviewées une description claire et complète du travail qu'ils réalisent. Ceci provient sans doute du fait des réflexes qu'elles acquièrent en accomplissant leur travail et du fait qu'il leur semble inutile d'expliciter ce qui est, pour eux, évident.

Dans [PEETERS, p 59], des indications nous sont fournies quant à la façon de procéder :

- utiliser les documents existants (organigrammes, règles de gestion telles que règles d'avancement ou de rémunération, notes internes, bordereaux, états, etc.) pour éclairer les résultats des interviews;
- recouper plusieurs interviews de personnes différentes effectuant en principe le même travail;
- faire interviewer plusieurs fois une même personne, si possible sans la lasser et en changeant d'"interviewer".

3.1.2. LES TRAITEMENTS

Sur base de ces interviews, nous avons tenté de dégager les activités et les tâches de l'existant, et ce, en suivant les critères d'identification proposés dans les définitions.

La particularité de notre application ne nous a pas permis de découper les actes de gestion conformément au canevas proposé (activités et tâches).

Aussi, nous avons choisi de définir une découpe plus représentative du système. Ainsi, nous avons pu mettre

en évidence un certain nombre de traitements relatifs à la gestion des documents et brochures.

Pour les documents, nous avons dégagé les différentes étapes de leur cycle de vie car notre objectif était de connaître de manière détaillée tous les éléments dont nous aurions à tenir compte dans le système futur. Si notre application avait été destinée à supporter la gestion d'une bibliothèque, nous n'aurions pas dû tenir compte des différentes étapes lors de l'élaboration d'un document mais plutôt de traitements du type "réception (enregistrement) d'un nouveau document". C'est de cette manière que nous avons procédé pour les brochures étant donné qu'elles proviennent de l'extérieur.

Une fois la découpe obtenue, nous pourrions nous interroger au sujet des critères d'identification des traitements. En effet, si l'on considère la finalité des traitements "gestion des documents" et "gestion des brochures", nous pouvons les recenser comme étant des activités. De même, pour les traitements tels que "rédaction d'un document", "traduction" ou "établissement d'une liste de distribution", on peut considérer qu'ils respectent les 3 unicités : temps, lieu et action et être ainsi recensés comme étant des tâches.

Cependant, il y a toujours moyen de discuter la découpe d'un système en tâches et activités. Ainsi, à la limite, on pourrait reprocher à la tâche de "rédaction" de ne pas correspondre exactement aux trois unicités puisqu'elle se déroule rarement sans interruption. Mais étant donné que c'est une tâche manuelle, ça n'a plus de raison d'être. D'autre part, on peut considérer que si le traitement respecte au moins un critère (minimum pour justifier une découpe) pour un niveau donné, sans répondre aux critères des autres niveaux, alors ce traitement peut être repris à ce niveau.

En considérant la façon dont nous avons procédé, on pourrait envisager de ne plus obliger le développeur à recenser les traitements selon certains critères liés à une terminologie, ces critères pouvant être sujets à interprétation, mais plutôt de lui laisser calquer la découpe à son type de problème. Le développeur pourra par la suite, essayer de se rattacher à la terminologie proposée. Dans tous les cas, le développeur doit s'attacher à justifier sa découpe. Il reste cependant à savoir jusqu'à quel niveau descendre dans la découpe pour l'étude de la situation actuelle. La solution serait de se référer à l'objectif de la découpe qui est de décrire ce qui est, afin de garder une trace de ce à quoi ressemblait l'existant avant le développement du projet. Dans le cas d'un système automatisé, on peut très vite décrire des détails inutiles. L'auteur de l'étude de l'existant devra alors estimer le niveau de détail de sa description. Par exemple, si le projet est

une modification d'un système existant, il peut être nécessaire d'approfondir seulement certains aspects de ce problème, c'est à dire ceux qui seront plus particulièrement touchés par les modifications.

Il ne faut pas oublier l'aspect spécification des traitements. Il n'est pas possible de donner une définition précise du contenu des spécifications. Elles doivent être rédigées en fonction de l'objectif assigné à l'étude de l'existant aussi, nous nous référons au bon sens du développeur pour l'accomplissement de cette tâche.

3.1.3. LES INFORMATIONS

A ce sujet, nous pensons qu'il y a un risque de confusion entre les différents types d'informations à recenser. Nous les classerons comme suit :

a) Les messages qui véhiculent un certain nombre d'informations et qui ont souvent une durée de vie limitée. Ces messages peuvent être écrits, oraux, externes (en provenance ou à destination de l'environnement) ou internes (circulant d'un traitement à l'autre). Exemples : bon de photocopie, demande de traduction, bon de commande. Remarquons qu'il existe aussi des messages implicites dans le cas, par exemple, d'une personne se donnant l'ordre de rédiger un document ou encore des messages artificiels tels qu'un message de fin de traitement. Ces derniers servent surtout d'événements déclenchants pour le traitement suivant.

b) Les données regroupées en collection qui consistent en une représentation des entités du réél perçu. Elles peuvent être créées, supprimées, modifiées ou consultées. Exemples : fichier des membres du personnel, ensemble des fiches bibliographiques.

c) Les objets physiques. Ce concept n'est pas vraiment utilisé dans la méthodologie de la CGER, cependant, on le retrouve au niveau du diagramme détaillé traitement-utilisateur (cfr. annexe A4). Il nous a semblé intéressant de les reprendre afin d'apprécier leur circulation de traitement en traitement. Aussi, nous avons attribué à chaque traitement le type de document qu'il peut recevoir en entrée et fournir en sortie pour le traitement suivant. Cette notion est d'autant plus importante pour nous puisque les objets physiques représentaient en quelque sorte leur propre description et tous les traitements (pour la gestion des documents) s'articulaient autour des objets physiques (les documents). A titre d'exemple, nous renvoyons le lecteur au chapitre précédent au point 4.

La confusion entre ces différentes informations n'est pas critique si les développeurs prennent le pli de décrire toutes les informations existantes en justifiant pourquoi ils veulent en tenir compte. Ils pourront ainsi y faire référence dans la description des traitements.

3.1.4. LA CONSTRUCTION DE LA REPRESENTATION DU SYSTEME ACTUEL

Nous avons rencontré beaucoup de difficultés à réaliser la représentaion schématique du système actuel de la gestion des documents. Ces difficultés proviennent surtout du fait de la multitude de scénarios possibles et des modifications des objets physiques selon ces scénarios, ce qui n'était pas le cas pour les brochures.

Il fallait d'abord mettre de l'ordre dans l'ordonnancement des traitements pour pouvoir ensuite le restituer sur un schéma. L'objectif du schéma étant de représenter au mieux la situation actuelle et de pouvoir le soumettre aux utilisateurs pour qu'ils indiquent si notre vue sur l'existant est correcte ou non.

Malheureusement, tout n'était pas représentable sur le schéma, c'est le cas pour les conditions de déclenchement de telle ou telle tâche. Si l'on considère la tâche "distribution", elle ne peut être effectuée que si le document existe à l'état dactylographié dans les deux langues. Nous avons donc veillé à reprendre à travers la description des éléments intervenant dans le schéma, ce qui n'était pas représenté. Aussi, celui-ci doit être interprété en regard avec sa documentation.

Nous pouvons conclure, à ce stade, que tout schéma doit être conçu en fonction de son objectif et aussi que plus le concepteur aura soigné sa documentation, plus l'interprétation faite par chacun des interlocuteur sera proche de la réalité perçue par le concepteur et ce n'est qu'à partir de ce moment là que les discussions se feront sur une base solide.

Dans notre cas, nous voulions représenter l'existant; les difficultés rencontrées à la construction du schéma nous ont permis de dégager les disfonctionnements du système actuel.

Il serait difficile de proposer une meilleure représentation car nous serions toujours limité par son formalisme. En effet, nous avons voulu envisager un autre type de modèle qui nous aurait permis de représenter l'enchaînement des traitements à l'aide de

structure de contrôle (point de décision, de synchronisation, parallélisme, itération). Nous nous sommes alors rendu compte que nous aurions été obligé de développer ainsi autant de schémas qu'il y avait de scénarios sans avoir la certitude d'avoir recouvert tous les cas. Ce genre de schéma est utile, selon nous, pour des applications orientées traitement où la filière d'exécution est rigoureusement définie (peu de scénarios possibles).

Il faut donc accorder une attention particulière à la documentation annexée au schéma car c'est là que l'on retrouvera toutes les informations nécessaires à sa compréhension. Le schéma est alors utilisé comme aide mémoire.

3.2. BESOINS, OBJECTIFS ET CONTRAINTES

Nous n'avons pas rencontré de problèmes particuliers pour l'établissement de ces éléments. Comme le montre l'application (cfr. chapitre 1), nous les avons décrit en texte libre. Nous pouvons cependant décrire un certain nombre d'observations.

3.2.1. LES BESOINS

Lors de la réalisation de l'étude de l'existant, certains besoins peuvent être dégagés surtout lorsqu'on aborde la partie critique. Cette dernière met, en effet, en exergue les déficiences qui peuvent conduire, de la sorte, à un besoin d'automatisation de certaines tâches, ou à la réorganisation des comportements de gestion ou des flux d'information.

Les conclusions de cette étape ne sont pas définitivese car les besoins pourront être redéfini au fur et à mesure du développement, comme, par exemple, au moment de l'analyse conceptuelle où les utilisateurs commencent à percevoir l'ampleur du système futur.

3.2.2. LES OBJECTIFS

Il ne faut pas confondre besoin et objectif. Les objectifs sont les fonctionnalités que le système devra remplir pour satisfaire les besoins des utilisateurs.

Afin d'aider les développeurs à déterminer ces objectifs, il serait intéressant qu'ils puissent disposer d'une liste non exhaustive d' objectifs qui

sont habituellement défini pour tout développement de projet. Ils pourraient exploiter cette liste, d'une part, en la complétant et, d'autre part, en pondérant ses éléments. Cette liste pourrait reprendre, par exemple, les objectifs informationnels décrit dans [BODART (a), pp 143-144] concernant la qualité de l'information (pertinence, lisibilité,...) et le comportement du système vis à vis de l'information (performance, sécurité,...) ou encore des objectifs stratégiques.

3.2.3. LES CONTRAINTES

Deux types de contraintes devrait être envisagés, les contraintes qu'il faut impérativement respecter telles que le délai de réalisation de chacune des phases de développement, l'enveloppe budgétaire, le lien avec d'autres d'applications et protocoles de communication avec d'autres entreprises [GRAAS (a), les contraintes internes] et les contraintes qu'il serait souhaitable de respecter tel que le temps de réponse.

Pour chacune de ces contraintes, le développeur pourrait définir certaines zones de valeurs selon le traitement (exemple : temps de réponse idéal < 2 secondes, temps de réponse refusé : + de 5 minutes).

C'est dans la mesure où les contraintes seront respectées que certains des objectifs seront réalisés. Exemple : prenons l'objectif de convivialité du système pour tel type de traitement interactif et les contraintes suivantes : temps de réponse raisonnable, taux de remplissage de l'écran < 60 %, 1 < nombre de couleurs <= 3 et utilisation de touches de navigation. Si le temps de réponse est de + ou - 5 minutes, l'objectif de convivialité pourrait ne pas être respecté en fonction du type d'application.

3.3. LES SOLUTIONS POSSIBLES

Comme nous l'avons expliqué au chapitre 1 concernant l'application, nous n'avons pas procédé à une analyse approfondie de toutes les solutions possibles. Notre but, ici, sera de proposer un processus de choix d'une manière plus systématique.

Nous considérons que le processus de choix devrait être vu comme un parcours dans un arbre où la racine représente l'identification du système à développer et chaque noeud représente une alternative de choix découlant de son ascendant direct.

En général, tout problème à une infinité de solution, aussi la construction de l'arbre de décision se base sur les principaux critères de choix qui sont mis en évidence au cours de réunions successives avec l'utilisateur ainsi que lors de négociation avec "les responsables de l'allocation des ressources" (ressources en matière de personnel, de matériel,...).

Le parcours dans l'arbre s'étale dans le temps et même au-delà de l'étape du choix d'une solution. Ce qui signifie qu'au cours du développement, de nouvelles alternatives vont se présenter. Ce qui implique qu'il faut garder une trace de tous les critères utilisés ainsi que des choix intermédiaires qui seront accompagnés de leur justification.

Cette façon de procéder permet de ne pas continuer inutilement dans une voie sans intérêt mais d'approfondir les alternatives intéressantes.

4. L'ANALYSE CONCEPTUELLE

Pour cette analyse, nous ne pouvons pas à proprement parler de difficultés de réalisation mais plutôt de constatations guidées par notre application et ce, aussi bien pour les données que pour les traitements.

4.1. L'ANALSE CONCEPTUELLE DES DONNEES

Le concepteur d'un schéma conceptuel des données peut être amené à donner des informations complémentaires pour la compréhension de son schéma dans le cas où il a jugé préférable de s'écarter de la sémantique du schéma.

Notre application décrite au chapitre 1, nous fournit une illustration de ce cas : la relation destinataire d'un document, DOC-PERS (DOCUMENT,PERSONNE) n'est pas identifiée par les identifiants des entités sur lesquels elle est définie. Or nous aurions pu ajouter à cette relation un identifiant propre tel que le moment précis de la création de la relation dans le système et ce, en vue de respecter la sémantique du schéma. Etant donné que nous ne voyons pas l'intérêt d'attribuer un tel identifiant à la relation, nous avons préférer déroger à la sémantique du schéma moyennant une contrainte de non intégrité supplémentaire.

4.2. L'ANALYSE CONCEPTUELLE DES TRAITEMENTS

Nous avons utilisé le même principe de découpe des traitements au niveau conceptuel qu'au niveau de l'étude de l'existant, aussi nous invitons le lecteur à se référer à l'étude de l'existant dans ce chapitre pour plus d'informations.

Nous allons cependant nous attarder un peu plus sur l'aspect des spécifications. En effet, nous avons pu observer une certaine redondance dans la spécification des différents traitements constituant notre application. Si nous reprenons la spécification du traitement "création d'une personne" (chpitre 1, p), on peut constater que, pour chacun des rôles assignés à une personne, le système vérifie l'existence de l'entité document, brochure ou groupe mise en relation avec la personne, il vérifie aussi que la création de cette relation n'entraîne pas d'incohérence par rapport au modèle conceptuelle des données décrit précédemment (exemple : création d'une personne responsable d'un document alors que ce document a déjà un responsable) et finalement il vérifie si la personne existe déjà ou non avant de la créer. Nous pourrions dans ce cas faire abstraction dans une certaine mesure des rôles qui pourraient être assigné à la personne

pour donner une spécification plus générale qui serait par exemple la suivante : création de la personne identifiée par un no-matricule donné (si la personne n'existe pas dans le système) avec les informations nécessaires la concernant (voir modèle conceptuel des données). Création de ses relations éventuelles suivant les rôles qui lui sont assignés (à condition que les entités correspondantes existent et en tenant compte des contraintes exprimées dans le modèle conceptuel des données) avec les informations nécessaires les concernant (voir modèle conceptuel des données).

Il faut donc veiller à alléger les spécifications lorsque cela s'avère possible, mais cela demande au développeur un effort d'abstraction, ce qui n'est pas toujours évident.

5. L'ANALYSE FONCTIONNELLE

Les difficultés que nous avons recontrées à ce niveau concernent surtout les quantifications et le dialogue utilisateur-système.

5.1. LES QUANTIFICATIONS

Les quantifications sont nécessaires pour l'élaboration de la base de données physique pour les aspects de performance. Cependant, ces quantifications ne sont pas toujours faciles à estimer mais dans notre cas c'est un moindre mal du fait du SGBD utilisé à savoir DB2/SQL. Aussi nous suggérons aux utilisateurs d'un tel SGBD d'effectuer ces quantifications en cours d'exploitation du logiciel. En effet, DB2/SQL permet d'optimiser les accès (par ajoût d'index au table) sans perturber les programmes. Cela se justifie d'autant plus que, dans notre cas, les accès à la base de données sont simples. Mais ceci n'est pas vrai pour tous les SGBD.

5.2. LE DIALOGUE UTILISATEUR-SYSTEME

Nous avons en fait réaliser le dialogue utilisateur-système d'une manière très détaillée car nous ne pensions pas avoir le temps d'implémenter notre application nous-mêmes. Par conséquent, on peut observer une certaine redondance dans les différents dialogues. Pour éviter ce problème, nous aurions pu l'élaborer en donnant un schéma de dialogue par classe d'opérations et expliciter les cas particuliers pour certains éléments de cette classe d'opérations. Nous aurions pu aussi regrouper les aspects conversationnel du dialogue (affichage de l'écran, presser ENTER, remplissage des champs,...) dans un exposé préliminaire et de ce fait laisser certaines choses implicites.

6. SYNTHESE

Nous pouvons conclure ce chapitre en nous interrogeant sur les raisons sous-jacentes aux problèmes que nous avons rencontrés.

Ils sont en général dus au manque d'informations pour la réalisation de certaines phases du développement comme par exemple les spécifications. En effet, la méthodologie énonce un ensemble de concepts mais ne donne pas toujours suffisamment de détail quand à la manière de les utiliser. D'autres problèmes proviennent de l'utilisation de modèles et de formalismes qui en général ne permettent pas une représentation adéquate et suffisamment explicite du réel perçu.

CHAPITRE 3

ELEMENTS DE SPECIFICATION D'OUTILS D'AIDE A LA DOCUMENTATION ET AU DEVELOPPEMENT SUR BASE DE L'APPLICATION "PUMA"

1. INTRODUCTION

Tout au long de ce chapitre, notre intention est de donner quelques indications quant aux spécifications d'un système automatisé qui aurait pu nous être utile pour le développement et la documentation de notre application.

Par système automatisé, nous entendons un ou plusieurs outils dont nous donnerons la spécification.
L'objectif que nous assignons à ce système est double. Premièrement, il doit servir de support à la fonction d'administration des données; en d'autres termes, son utilisation doit permettre de "mettre de l'ordre" dans tous les éléments qui interviennent dans le développement et la maintenance de projets (analyses, données, traitements, personnes, objets physiques,...) et ce, dans le cadre d'une démarche méthodologique.
Deuxièmement, on attend du système qu'il puisse accélérer dans une certaine mesure certaines phases du développement.

Pour la clarté de l'exposé, nous adopterons une démarche en deux temps. Tout d'abord, nous supposerons que nous disposons d'un système automatisé de documentation "idéal" (SAD), en ce sens qu'il répond à tous les besoins exprimés par l'utilisateur (écran graphique, dialogue convivial, création de vues propres à chaque utilisateur, gestion des identifiants,...). Nous donnerons un aperçu des caractéristiques que nous attendons de ce SAD.
Ensuite, nous donnerons des indications quant à la manière dont ces caractéristiques pourraient être attribuées au dictionnaire DATAMANAGER, en supposant qu'il dispose d'une interface avec un outil comme EXCELERATOR, outils décrits dans le deuxième chapitre de la première partie.

Dans notre étude, nous ne nous baserons que sur l'application que nous avons développée (dont la description se trouve au premier chapitre de cette même partie). Cependant, de temps à autre, nous proposerons des fonctions plus générales de manière à conférer une certaine flexibilité au SAD.

Nous n'aborderons pas un point essentiel qui consiste à prévoir la "migration" vers notre SAD des données de l'organisation actuellement documentées sous une forme ou une autre (manuelle ou automatique).
Nous adopterons donc une démarche de type "cleaning up the mess" en ce sens que le SAD ne s'applique qu'aux nouveaux systèmes d'informations, exactement comme si la naissance du SAD était conjointe à celle de l'organisation.

Remarquons en dernier lieu que nous n'avons pas pu faire preuve d'exhaustivité et de rigueur dans la spécification du SAD car cela exigeait une étude beaucoup plus approfondie. La complexité du problème et le temps qui nous était imparti ne nous ont malheureusement pas permis de mener à bien une telle étude. Nous espérons cependant que les problèmes soulevés pourront constituer une ébauche pour une recherche ultérieure.

2. SPECIFICATION D'UN SYSTEME AUTOMATISE DE DOCUMENTATION "IDEAL"

La spécification du SAD se déroulera en cinq étapes.

Tout d'abord, nous recenserons les informations qui, selon nous, doivent être intégrées au SAD pour qu'il puisse satisfaire aux objectifs qui lui sont assignés. Nous classerons ces informations suivant la phase du cycle de vie à laquelle elles sont relatives. A ce niveau, les seules relations que nous considérerons entre ces informations sont des relations que nous appellerons "structurelles". Ces relations sont limitées à la fois à la phase du cycle de vie documenté et aux types d'éléments du système d'information pour cette phase : données, traitements, écrans,... Ces relations uniront par exemple des données de la phase conceptuelle, des traitements de la phase technique,...

Dans un deuxième point, nous décrirons toutes les relations "inter-phases" utiles, ainsi que les relations entre les différents types d'éléments d'une même phase (relations "inter-structurelles").

Ensuite, nous donnerons quelques exemples de contrôles que le SAD pourrait effectuer sur toutes ces informations.

Finalement, nous suggérerons un mode de dialogue destiné à guider l'utilisateur du SAD et nous conclurons par quelques idées de traitements automatiques susceptibles de préparer la documentation des phases ultérieures du développement.

Remarques générales

1. Nous attendons du SAD qu'il se compose d'un ensemble de membres entre lesquels il existe un éventail de relations; toute relation peut être parcourue dans les deux sens.

2. Le SAD dispose d'un écran graphique convivial par l'intermédiaire duquel on peut documenter les différents membres.

Le SAD est un SGBD sur lequel sont notamment greffés deux utilitaires qui utilisent la base de données commune à tout le système :

- un utilitaire "graphe de documentation",
- un utilitaire "aide à l'élaboration de modèles".

Ces utilitaires seront décrits au travers des points suivants.

2.1. Recensement des informations à intégrer au SAD

Ces informations ont pour objectif de documenter le développement du projet. Vu la complexité de la tâche de documentation, nous pensons qu'il n'est pas souhaitable que le SAD renferme la totalité des dossiers de documentation. Si tel était le cas, nous n'aurions pas les moyens de limiter les informations à introduire dans le SAD qui, à long terme, aurait de grandes chances de contenir de la "littérature" trop abondante qui encombrerait le système et découragerait ses utilisateurs.

Nous voyons plutôt le SAD comme un outil destiné à guider les personnes en quête de certains renseignements au sujet d'un projet. Ceci implique qu'il existera toujours des dossiers de documentation. Il est évident qu'en procédant ainsi, beaucoup d'informations seront redondantes dans l'organisation, c'est pourquoi il est indispensable de limiter la description des éléments du S.I. aux données nécessaires et suffisantes à la poursuite de notre objectif double. Cet objectif, selon nous, ne serait pas aisément réalisable si l'on s'en tenait à un système de documentation manuelle via la production de dossiers.

Avant de recenser les informations par phase du cycle de vie d'un projet, nous allons décrire en quoi consistent les utilitaires "graphe de documentation" et "aide à l'élaboration de modèles".

2.1.1. L'utilitaire graphe de documentation

Cet utilitaire permet de définir un graphe de documentation qui sera la "charpente" sur laquelle vont se greffer toutes les informations relatives au projet.

Si la démarche méthodologique se base sur des phases bien délimitées, il est possible de formaliser ce graphe dont le but est de "documenter la documentation" du projet. (voir figure 1)

GRAPHE DE DOCUMENTATION

Niveau 1 MG1 Niveau 2 MG2 Niveau 3 MG3.

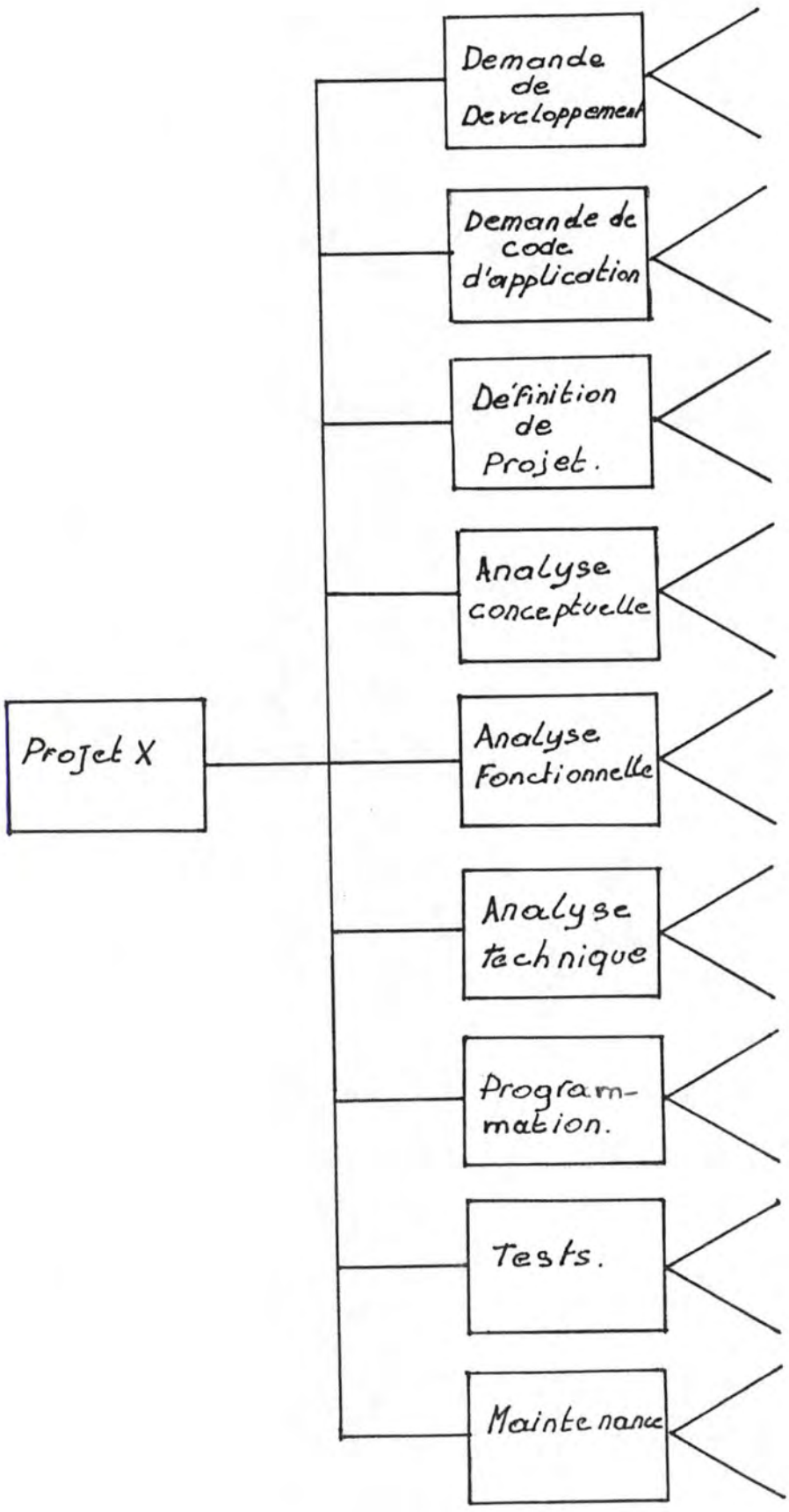


figure 1.

Ce système peut proposer une hiérarchie de membres où, au plus haut niveau, un membre décrit succinctement le projet et est relié à des membres de niveau 2 associés chacun à une phase du cycle de vie du projet. Chaque membre de niveau 2 décrit tous les éléments constituant la documentation relative à cette phase (référence à un dossier ou document, référence à des informations reprises dans un autre outil de documentation ou référence à un modèle ou à d'autres éléments décrits dans le SAD (dans ce dernier cas, la référence peut être exprimée par une relation dans le SAD)).

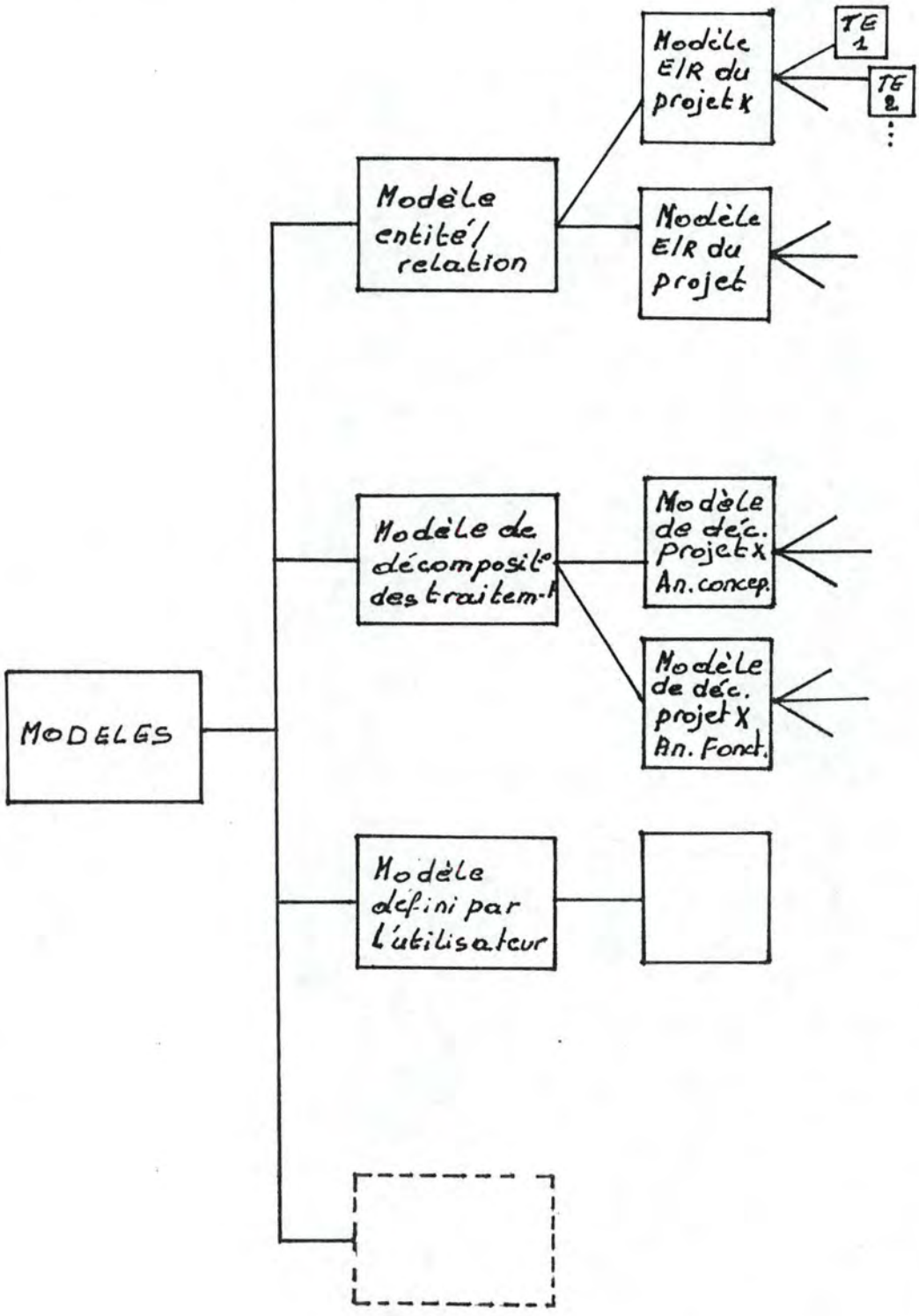
Dans le but de faciliter la gestion du projet, il serait intéressant que chaque membre de niveau 2 contienne pour chaque référence la date de la dernière mise à jour ainsi que le nom de la personne qui est responsable de cette mise à jour.

2.1.2. L'utilitaire aide à l'élaboration de modèles

La documentation des différentes phases fait quelquefois appel à des modèles. Parmi les modèles utilisés, certains sont reconnus au niveau de tout le département de développement de projets (par exemple : le modèle entité/relation, le modèle de décomposition des traitements,...). Par contre, il se peut que le développeur ressente le besoin d'utiliser un autre modèle plus approprié à son problème.

C'est dans cette hypothèse que l'utilitaire modèles trouve sa raison d'être puisque son objectif est de permettre de décrire et de réaliser des modèles "sur mesure". (voir figure 2)

AIDE A L'ELABORATION DE MODELES



Niveau 1	Niveau 2	Niveau 3	Niveau 4
Répertoire des modèles.	Descriptions des modèles.	Occurrences des modèles	Constituants des occurrences des modèles.
MM1.	MM2.	MM3.	

figure 2.

Considérons un membre de niveau 1 MODELES dont l'unique fonction est de répertorier les noms des types de modèles utilisés. Parmi ces modèles, les plus courants peuvent être prédéfinis; nous pensons au modèle entité/relation, au diagramme des flux (Gane & Sarson), au schéma HIPO (Hierarchical Input Process Output),... D'autres peuvent être définis par l'utilisateur selon ses propres besoins.

De manière générale, la définition d'un type de modèle consistera à intégrer sa sémantique dans un membre de niveau 2 représentant le modèle et qui est associé au membre de niveau 1 MODELES.
Ce membre de niveau 2 doit également comprendre la signification de chacun des symboles utilisés (s'il s'agit d'un schéma). Cette description doit être rédigée en texte libre à l'intention de l'utilisateur.

Par ailleurs, nous pouvons supposer que le SAD fournit un langage à l'aide duquel on peut définir la manière dont le système doit comprendre les différents symboles introduits à l'écran. Ainsi, il pourra exécuter un certain nombre de traitements en conséquence ; par exemple, si tel symbole est dessiné à l'écran, le système devra demander à l'utilisateur d'introduire sa description, puis il devra l'enregistrer dans un membre d'un type bien précis. Dans ces conditions, on peut également considérer que l'utilisateur peut créer lui-même ses propres icônes.

Une fois le modèle décrit, l'utilisateur peut donc créer une occurrence du modèle (schéma attaché à telle phase de tel projet). Une occurrence d'un modèle est décrite dans un nouveau membre de niveau 3 qui sera relié de manière appropriée au graphe de documentation. Ce membre contiendra éventuellement du texte libre justifiant le choix du modèle ainsi que quelques commentaires (par exemple, une référence vers un dossier contenant les contraintes d'intégrité pour un modèle entité/relation). Ce membre sera également mis en relation avec d'autres membres définissant les constituants du schéma et leurs relations (membres de niveau 4). Ainsi, nous aurions par exemple, pour le modèle entité/relation, des membres de type entité, relation, attribut,... et des relations de type identifie, fait partie de,...

2.1.3. Documentation d'un projet

Nous ferons ici quelquefois référence à des membres gérés par les utilitaires "graphe de documentation" et "aide à l'élaboration de modèles". Nous les noterons d'une manière générale :

- MGi pour membre de niveau i de l'utilitaire graphe de documentation;

- MMi pour membre de niveau i de l'utilitaire aide à l'élaboration de modèles.

Dans ce point, nous reprendrons les informations que nous souhaiterions pouvoir intégrer dans le SAD pour chacune des phases du cycle de vie d'un projet.

a) La définition de projet

Le MG2 relatif à cette phase fera référence :

- à des dossiers de documentation pour les étapes suivantes :
  - description de la situation actuelle en texte libre,
  - définition des objectifs, besoins et contraintes du nouveau système,
  - critique de l'existant,
  - autres dossiers éventuels;
- à une ou plusieurs occurrences de modèles, c'est-à-dire à un ou plusieurs MM3;
- à un point d'entrée pour le choix d'une solution si l'on utilise la technique décrite ci-dessous.

En ce qui concerne les solutions possibles, nous invitons le lecteur à se référer au point concernant le développement de l'application PUMA face à la méthodologie. Nous y avions décrit le processus de choix comme étant un parcours dans un arbre. Ce parcours s'étale dans le temps même au-delà de l'étape du choix d'une solution. En effet, d'une part, avant de choisir une solution, les principaux critères de choix sont mis en évidence au cours de réunions successives avec l'utilisateur ainsi que lors de négociations avec "les responsables de l'allocation des ressources", ressources en matière de personnel, de matériel,.... D'autre part, une fois une solution choisie à ce niveau, les négociations ne sont pas terminées car, au cours de tout le développement, de nouvelles alternatives vont se présenter. Ceci implique qu'il faut garder une trace des critères, des choix intermédiaires qui seront effectués et de leur justification.

Il serait intéressant que le SAD puisse supporter ce processus en représentant l'arbre de décision par une hiérarchie de membres.
Ainsi, un membre décrivant un noeud contiendrait : la description du critère associé au noeud, la description des alternatives et, pour chaque alternative, la description de ses avantages et inconvénients ainsi que la solution choisie (ce qui s'accompagnerait de la création d'un nouveau membre pour le critère (ou noeud) suivant).

En définitive, ne serait décrit que le parcours dans l'arbre et non pas l'ensemble des feuilles de l'arbre (toutes les solutions "possibles").

b) L'analyse conceptuelle

Le MG2 relatif à cette phase fera référence à plusieurs modèles.

Pour les données, nous garderons le modèle entité/relation où le MM3 contiendra une référence à un dossier contenant les contraintes d'intégrité. Nous pensons également inclure les quantifications statiques à ce niveau pour les types d'entités et types de relations.

Pour les traitements, nous utiliserons le modèle de décomposition. Se pose alors le problème de la spécification des traitements.

Dans le cas des traitements batch ou des traitements pour lesquels les données en entrée, en sortie et mises à jour sont bien définies, nous suggérons une description de type HIPO pour chaque traitement. Si la spécification est trop complexe, il doit être possible de donner la référence d'un dossier d'accompagnement du projet.

Dans le cas de traitements interactifs, tels que ceux que nous avons dégagés dans notre application, nous pensons qu'un formalisme de ce type n'est pas adéquat. Le MM3 relatif à la décomposition des traitements devrait contenir la référence du dossier où sont incluses les spécifications des traitements. Néanmoins, pour répondre à l'objectif d'administration des données, il est souhaitable que chacun des traitements soit identifié dans le SAD (par son nom et une brève description).

c) L'analyse fonctionnelle

En ce qui concerne les données, nous avons reporté les quantifications statiques à l'analyse conceptuelle.

En ce qui concerne les traitements, nous proposons un nouveau modèle de décomposition des traitements pour chacun des traitements dégagés lors de l'analyse conceptuelle. Dans notre cas, il y a une relation 1-1 entre les traitements de l'analyse conceptuelle et ceux de l'analyse fonctionnelle mais cela n'est pas le cas pour la majorité des projets.

Pour chacun des traitements fonctionnels, nous mentionnerons une fréquence d'exécution.

A chacun des traitements terminaux (transactions), nous associerons des membres destinés à spécifier les transactions de type base de données. Remarquons que, dans notre application, les transactions utilisateurs sont les traitements du niveau conceptuel puisque l'application est totalement interactive.
Au sujet des transactions, nous pensons que le MG2 relatif à l'analyse fonctionnelle pourrait faire référence à un MG3 globalisant des informations communes

à toutes les transactions du projet. Ce membre MG3 serait mis en relation avec chacun des membres décrivant les transactions.

En ce qui concerne le dialogue utilisateur/système pour chaque traitement du niveau fonctionnel, nous pensons qu'il est préférable de le garder dans un dossier auquel il sera fait référence dans la description du traitement.

En ce qui concerne les écrans, le MG2 relatif à la phase fonctionnelle fera référence à un MG3 de type INTERFACE qui contient la liste des noms de tous les écrans du projet et qui est relié à autant de membres (de niveau 4) qu'il y a d'écrans. Ces membres renfermeront : le nom, la description et la référence APE de l'écran.

d) L'analyse technique

En ce qui concerne les données, un formalisme peut être adopté pour la documentation du modèle conforme au SGBD DB2. Le MG2 peut faire référence à un membre de type SCHEMA décrivant le schéma conforme au SGBD. Ce membre sera relié à des membres de type TABLE eux-mêmes reliés à des membres de type CHAMP qui décriront chacune des colonnes d'une table.

En ce qui concerne les traitements, le MG2 peut aussi faire référence à un membre de type SYSTEME relié à un ensemble de membres de type PROCEDURE dans lesquels nous spécifierons chacune des fonctions APL. A ce niveau, un formalisme de description pourrait également être adopté.
Nous spécifierons de la même manière les requêtes SQL. Dans ce cas, un membre MG2 peut faire référence à un membre de type ACCESBD contenant le libellé de chacune des requêtes.

e) La programmation

Nous prévoyons dans le MG2 une référence à l'application PUMA et à un dossier d'accompagnement ainsi qu'une référence technique au schéma SGBD implémenté.

f) Les tests

Le MG2 prévoira une référence vers un dossier contenant les jeux de test.

g) Maintenance

Dans le cas (trop fréquent) où les mises à jour de la documentation initiale ne sont pas réalisées, le MG2 peut contenir une référence vers des dossiers de maintenance.
Le système pourrait également gérer des versions successives de la documentation d'un projet. Dans ce

cas, le MG2 attaché à la maintenance contiendrait des informations au sujet de l'état des différentes versions.

2.2. LES RELATIONS INTER-STRUCTURELLES ET INTER-PHASES

Jusqu'à présent, nous n'avons considéré que les relations structurelles.

Ces relations sont :

- au niveau du graphe de documentation, des relations :
  - . du membre définition de projet vers des membres de niveau 2 relatifs aux phases du cycle de vie d'un projet,
  - . d'un membre de niveau 2 vers des modèles ou des membres de type interface, accèsbd, schéma ou système;
- au niveau des modèles, des relations :
  - . d'un membre de type MODELES vers des membres de description des modèles (niveau 2),
  - . d'un membre de description d'un modèle vers des occurrences de ce modèle (niveau 3),
  - . d'un membre occurrence d'un modèle vers des membres décrivant les éléments constituants le modèle (niveau 4);
- au sein de chaque phase, des relations limitées aux relations entre les éléments constitutifs des différents modèles réalisés.

Ci-dessous, nous définirons les relations inter-structurelles pour les éléments d'une même phase, c'est-à-dire les relations entre les éléments de nature différente appartenant à une même phase du développement; ensuite, nous définirons les relations inter-phases qui permettent d'établir des "mappings" entre les membres (éventuellement de nature différente) n'appartenant pas à la même phase.

2.2.1. LES RELATIONS INTER-STRUCTURELLES

a) Pour l'analyse conceptuelle

Nous prévoyons les relations suivantes :

- des relations du type utilise entre les traitements et les types d'entités et de relations (dans le cas ou le modèle entité/relation a été choisi) ou entre les traitements et une autre représentation des données (dans le cas contraire);

- des relations de type confère entre différents traitements à fort degré de couplage. Dans ce cas, une modification apportée à un traitement a de fortes chances d'entraîner des modifications dans les traitements associés. L'intérêt de ce type de relation est donc de permettre de connaître tous les traitements susceptibles de devoir être modifiés suite à la mise à jour de certaines informations.

b) Pour l'analyse fonctionnelle

Nous prévoyons des relations du type utilise entre les traitements et les transactions de type base de données ainsi qu'entre les traitements et les écrans.

c) Pour l'analyse technique

Nous suggérons les relations suivantes :

- des relations du type utilise entre les procédures APL et les requêtes SQL;

- des relations du type utilise entre les requêtes SQL et les tables DB2.

2.2.2. LES RELATIONS INTER-PHASES

a) L'analyse conceptuelle et l'analyse technique

Nous proposons les relations suivantes :

- des relations du type engendre entre les traitements conceptuels et les traitements fonctionnels;

- des relations de type est utilisé par entre les types d'entités et de relations et les écrans dont les champs correspondent aux attributs de ces types d'entités et de relations.

b) L'analyse conceptuelle et l'analyse technique

Nous prévoyons des relations de type engendre entre les types d'entités et de relations et les tables DB2.

c) L'analyse fonctionnelle et l'analyse technique

Nous proposons les relations suivantes :

- des relations de type engendre entre les transactions de type base de données et les requêtes SQL;

- des relations de type est utilisé par entre les écrans et les procédures APL.

2.3. LES CONTROLES

Nous envisageons ici une série de contrôles les plus courants :

- les contrôles syntaxiques qui sont d'autant plus complexes que le système est flexible;

- des contrôles de l'unicité de l'identifiant et, si possible, de l'alias ou d'un groupe d'alias.
  Nous pensons à ce sujet qu'il est préférable d'utiliser une concaténation d'alias comme identifiant plutôt que d'utiliser un libellé non significatif. La portée d'une concaténation d'alias pourrait être limitée à un contexte bien déterminé, celui du projet par exemple.

Outre ces contrôles généraux, nous prévoyons une série de vérifications que nous examinons successivement ci-dessous.

2.3.1. LES CONTROLES LIES A LA DOCUMENTATION GLOBALE

Cette documentation porte sur tous les projets. Dans ce cadre, le SAD peut avertir l'utilisateur au cas où un membre n'est relié à aucun autre (et n'est pas un membre de type définition de projet du graphe de documentation) ou dans le cas où une phase d'un projet ne contient aucune référence ou encore pour toutes les situations susceptibles de traduire un oubli ou une erreur.

2.3.2. LES CONTROLES LIES AUX MODELES

Outre les contrôles syntaxiques éventuels, on peut imaginer des contrôles du respect de la sémantique d'un modèle. Par exemple, lorsque l'utilisateur veut valider un schéma entité/association, le SAD peut effectuer un certain nombre de contrôles. Il peut vérifier si tous les éléments ont été spécifiés, si toute entité a son identifiant, si tous les couples (type d'entité, type de relation) ont une connectivité,...

Selon nous, ces contrôles ne doivent pas être bloquants. De plus, l'utilisateur doit pouvoir prendre l'initiative d'activer ou non ces contrôles dans sa session de travail.

2.3.3. LES CONTROLES LIES AUX RELATIONS INTER-STRUCTURELLES

Pour les phases conceptuelle, fonctionnelle et technique, le SAD peut déceler, par exemple, s'il existe des traitements n'utilisant aucune donnée.

2.3.4. LES CONTROLES LIES AUX RELATIONS INTER-PHASES

Le SAD peut vérifier s'il manque des relations de "mapping" entre deux phases et ce, sur base d'un ensemble de règles prédéfinies, règles propres au type de projet et au type de modèles utilisés.

2.4. TYPE DE DIALOGUE UTILISATEUR/SYSTEME

Il serait intéressant que le dialogue se déroule sur un écran graphique multi-fenêtres (écran de type "table de travail") où, pour chaque élément activé (à l'aide d'une souris, par exemple), le système propose à l'utilisateur une série de fonctions.

Nous pensons par ailleurs que le SAD devrait fournir un utilitaire permettant d'adapter le dialogue aux besoins de l'entreprise. Ceci aurait un intérêt notamment dans le cas où l'on voudrait proposer à l'utilisateur tout un ensemble de contrôles à exécuter a posteriori sur les membres de la base de données du SAD. En adaptant le dialogue, on gardera une interface homme/machine uniforme pour tout le SAD.

Nous n'avons pas l'intention de spécifier le dialogue (ce qui ferait l'objet de tout un mémoire) mais de donner à titre d'exemple un scénario reflétant la manière dont nous souhaiterions pouvoir converser avec le SAD.

Le scénario qui suit a trait à la manipulation de données, nous avons décrit un dialogue utilisateur/système pour la création d'un schéma conceptuel des données où l'utilisateur modifie les icônes proposées par le système.

UTILISATEUR	SYSTEME
	- demander le nom du projet.
- introduire le nom du projet.	
	- présenter le graphe de documentation du projet.
- choisir la phase à documenter.	
	- présenter un canevas permettant à l'utilisateur : - d'introduire des références à des dossiers, - de réaliser ou de compléter un modèle, - de documenter un membre.
- choix : réaliser un modèle des données.	
	- afficher l'ensemble des types de modèles.
- choix : modèle E/R.	
	- proposer l'ensemble des icônes du modèle.
- modifier des icônes.	
	- établir la correspondance entre les icônes initiales et les icônes redéfinies.
- dessiner le schéma. - valider le schéma.	
	- avertissement : des éléments n'ont pas été décrits.
- décrire quelques éléments du schéma. - valider le schéma en spécifiant que des éléments resteront sans description.	
	- créer dans la BD les membres et les relations traduisant les éléments du schéma. - effectuer des contrôles syntaxiques et sémantiques.
- correction du schéma. - ...	

Une fois le schéma correct, le système peut proposer à l'utilisateur de créer des relations inter-structurelles ou inter-phases avec d'autres éléments du même projet ou d'un autre projet.

Un dialogue de sélection de données pourrait se dérouler selon le même principe : l'utilisateur naviguerait dans la documentation du projet et pourrait demander pour un élément donné la liste des relations structurelles, inter-structurelles ou inter-phases selon certains critères de sélection.

2.5. LES TRAITEMENTS AUTOMATIQUES

Des traitements automatiques pourraient alléger la tâche de l'utilisateur dans l'introduction des descriptions. Ainsi, par exemple, lorsque l'utilisateur documente un modèle de décomposition des traitements, le SAD pourrait dessiner pour un niveau donné un ensemble de boites destinées à accueillir la description des traitements du niveau inférieur.

Le système pourrait également prendre l'initiative de proposer à l'utilisateur de créer des relations entre des traitements et des éléments du schéma conceptuel des données d'un même projet.

Il serait également intéressant d'envisager que le SAD dispose d'une interface dynamique avec un outil de développement (APE dans notre cas). Cette interface permettrait au SAD de charger lui-même sa base de données à partir, par exemple, des panels qui ont été définis par APE. Il produirait ainsi pour chacun des panels un membre de type PANEL contenant son nom, la description de chacun de ses champs, ... Si c'est possible, le SAD pourrait également mettre en relation ces champs avec les données DB2 qui y correspondent et ce, sur base de tables de correspondance définies par l'utilisateur.

Il est possible d'imaginer un grand nombre de traitements automatiques de ce type facilitant la tâche de l'utilisateur.

2.6. REMARQUES GENERALES

a) Le principe de base des spécifications du SAD est qu'il constitue un outil flexible de documentation. En effet, il met à la disposition de l'utilisateur un ensemble extensible de techniques parmi lesquelles il peut effectuer un choix. L'utilisateur pourra ainsi documenter son projet "sur mesure", le graphe de documentation actant la démarche adoptée.

b) Un système permettant à l'utilisateur de spécifier lui-même ses propres modèles n'est pas très réaliste car il suppose un gros effort d'abstraction pour la tâche de

spécification d'un modèle. Par ailleurs, pour que le SAD puisse interpréter les icônes du nouveau modèle, il faut pouvoir manipuler un langage de commandes très complexe lié notamment à l'interface graphique.
Par ailleurs, il faut remarquer que ce n'est pas parce que l'utilisateur a la possibilité de définir tous les modèles imaginables qu'il arrivera toujours à spécifier sans peine toutes les informations qu'il souhaite.
Pour remédier partiellement à ce problème, nous prévoyons toujours une clause "commentaires" pour chaque modèle, où l'utilisateur pourra donner une justification du choix du modèle et décrire la manière dont ce dernier doit être interprété.
Si aucun type de représentation ne lui convient, l'utilisateur a la possibilité de faire référence à un dossier de documentation.

c) En ce qui concerne les traitements automatiques, ils ne seront efficaces que s'ils se basent sur des formalismes compréhensibles par la machine. Avant d'envisager tout traitement de ce type, il faut donc pouvoir évaluer si l'on dispose bien d'un formalisme qui ne constituera pas une contrainte pour l'utilisateur, contrainte en ce sens où ce dernier ne pourra pas représenter exactement ce qu'il désire à l'aide du formalisme proposé. Dans le cas contraire, les informations que le système pourrait générer automatiquement risquent de ne pas être satisfaisante aux yeux de l'utilisateur car déduites de données biaisées par rapport à ce qu'il voulait exprimer.

3. TRANSPOSITION DES CARACTERISTIQUES DU SAD A DATAMANAGER ET EXCELERATOR

Nous supposerons dans ce paragraphe qu'il existe une interface entre le dictionnaire DATAMANAGER (DMR) et EXCELERATOR (XL), interface qui est disponible sur le marché. Nous rappelons que ces deux outils ont été décrits au chapitre 2 de la première partie.

Notre objectif est d'examiner dans quelle mesure cette combinaison d'outils (DMR-XL) peut réaliser les fonctions que nous avons spécifiées ci-dessus pour le SAD.

Pour plus de clarté, nous reprendrons le même type de découpe que dans le second paragraphe.

3.1. LE GRAPHE DE DOCUMENTATION

Ce graphe peut être géré par DMR, ce dernier devrait mettre à la disposition de l'utilisateur une structure constituée comme suit :

- un membre MG1 de type DOCUMENTATION,
- plusieurs membres MG2 de type PHASE,
- des membres MG3 :
  - de type OCCMOD pour une occurrence d'un modèle,
  - de type SYSTEME pour l'identification du système opérationnel,
  - de type ACCESBD regroupant l'identification de tous les modules d'accès,
  - de type INTERFACE regroupant l'identification de tous les écrans,
  - de type SCHEMA pour un schéma SGBD,
  - ...

Un membre DOCUMENTATION sera associé à un membre de type PHASE via une relation CONTAINS.
Un membre de type PHASE sera associé à un des membres MG3 via une relation de type SEE car cette dernière peut relier des données et des traitements entre eux.

Dans les clauses descriptives de chacun de ces membres, il existe une clause REFERENCE que l'on exploitera pour mentionner un ou plusieurs dossiers d'accompagnement, les références à un programme ou à une analyse effectuée éventuellement à l'aide d'un autre outil (pour lequel DMR ne dispose pas nécessairement d'une interface).

Au travers des points suivants, nous retrouverons successivement les différents membres MG3.

3.2. LA REALISATION DE MODELES

A ce sujet , nous avions prévu pour le SAD un membre de type MODELES (MM1), des membres de description d'un modèle que nous appellerons DESCRMOD (MM2) ainsi que des membres de type OCCMOD (MM3) identifiant une occurrence d'un modèle donné.

Dans DMR, nous retrouvons les mêmes types de membres. Le membre MODELES (DESCRMOD) sera associé à un membre DESCRMOD (OCCMOD) via une relation CONTAINS.

C'est au niveau des membres de type OCCMOD qu'intervient XL. En effet, une interface permet d'importer des fichiers à partir de ou vers un autre dictionnaire comme DMR. XL possède donc son propre dictionnaire de données dont la structure est reprise figure 3.

STRUCTURE DU DICTIONNAIRE D'EXCELERATOR.

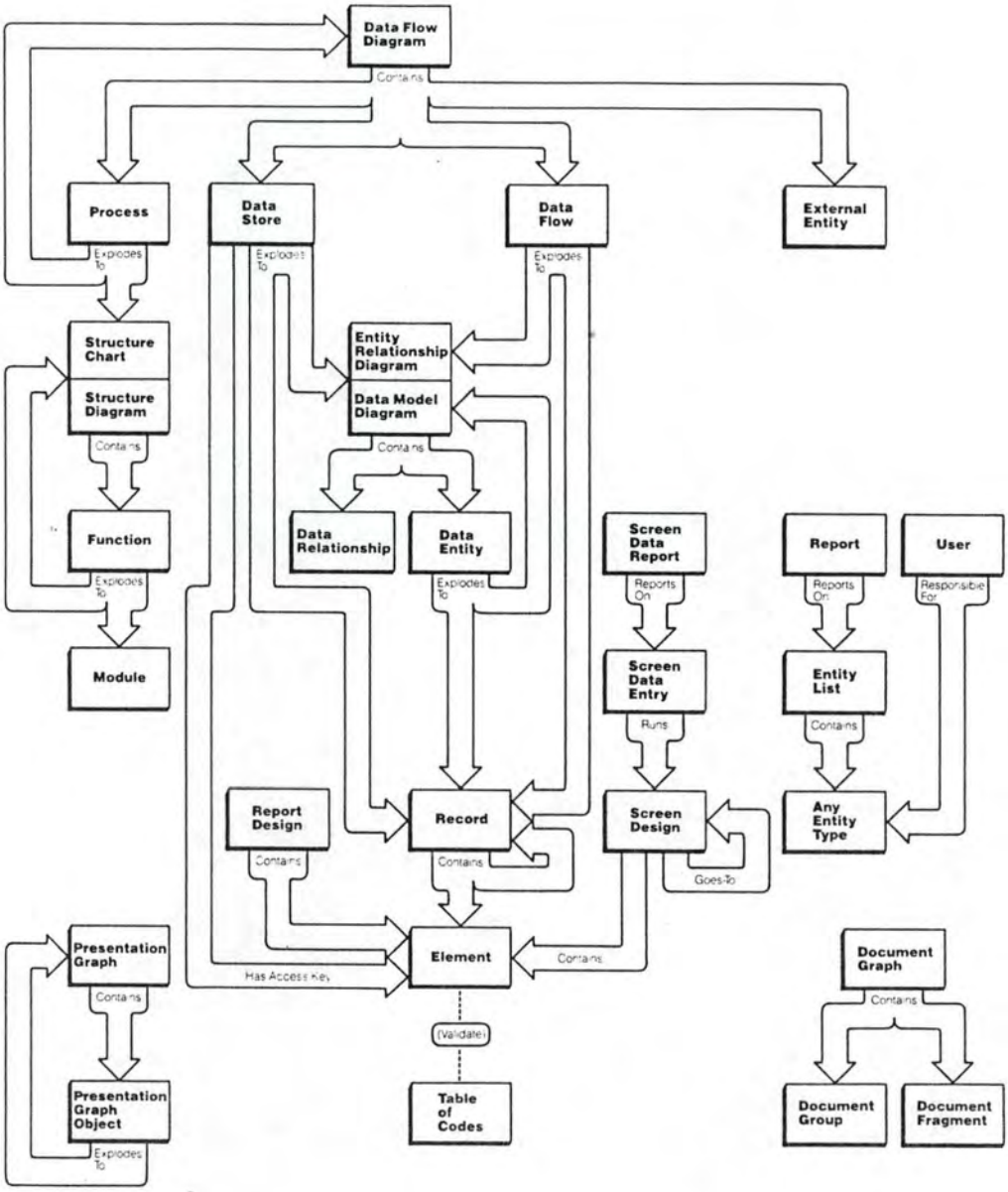


figure 3.

Ces fichiers transférés contiennent des membres définis via XL et nous supposons qu'il est possible de transférer dans DMR toutes les informations relatives à un schéma (par exemple des entités : Data Model Diagram, Data Relationship, Data Relationship, Data Entity, Record, Element ainsi que des relations existant entre elles, tous ces éléments représentant un modèle des données).

Dans le système DMR-XL, nous avons choisi de considérer DMR comme "master" étant donné la multitude de fonctions supplémentaires qu'il propose (langage de commandes élaboré, contrôles et syntaxe définis partiellement par l'utilisateur, nombreux utilitaires, outil mainframe,...). L'interface entre les deux outils devra être dynamique en ce sens qu'elle doit assurer la répercussion des modifications effectuées de part et d'autre sur des données commmunes.

Il est cependant possible de construire des schémas et de documenter un projet uniquement par XL vu qu'il dispose de son propre dictionnaire. Ceci pourrait être exploité dans le sens où XL servirait pour la conception d'un schéma et où DMR, une fois les informations importées, effectuerait une série de contrôles de validité et de cohérence.

Nous allons examiner de quelle manière DMR peut accueillir les informations relatives à un modèle.
Nous avons défini un type de membre OCCMOD attaché au type de membre DESCRMOD correspondant. L'importation d'un modèle défini via XL entraînera la création automatique d'un membre de type OCCMOD, représentant une occurrence de ce modèle, ainsi que d'une série de membres correspondant aux constituants du modèle importé.

Une fois le modèle importé, l'utilisateur pourrait, via DMR, créer toute une série de relations :

- entre le membre OCCMOD et le graphe de documentation,
- entre les membres constituants du schéma vers d'autres membres du DD (des traitements, par exemple) via une relation de type SEE (car il n'exite pas de relation USE).
- une série de relations inter-structurelles et inter-phases.

Ces relations doivent évidemment rester transparentes à XL.

Dans ces conditions, XL serait utilisé pour l'élaboration graphique de schémas (et donc sous-utilisé) et DMR couvrirait tout l'aspect d'administration des données.

Remarquons finalement que le système DMR-XL ne permet pas à l'utilisateur de définir ses propres modèles, ce qui constitue une limite à nos spécifications initiales.

3.3. LA DOCUMENTATION

Nous traiterons simultanément les types de membres SYSTEME, ACCESBD et INTERFACE en analysant en particulier le type de membre SYSTEME destiné à décrire l'ensemble des programmes du système. Dans ce cas, XL n'est pas utilisé, la documentation repose uniquement sut DMR.

Il existe une relation de type SEE entre un membre de type PHASE du graphe de documentation et le membre SYSTEME. Le membre SYSTEME est relié à des membres de type PROCEDURE via une relation de type CONTAINS.

DMR permettra à l'utilisateur via son langage de commandes de documenter ces membres et de créer explicitement des relations entre ces derniers.

3.4. LES CONTROLES

Ils peuvent être effectués par DMR moyennant une analyse approfondie de tous les types de membres existant dans le DD ainsi que des relations qui peuvent exister entre eux. Comme pour le SAD, il est possible d'imaginer tout un éventail de contrôles.

Notons à ce sujet que DMR peut effectuer des contrôles définis par le responsable du dictionnaire, contrôles qui devront s'exécuter à chaque introduction d'un nouveau membre. Par ailleurs, il peut fournir un certain nombre de membres à un programme d'application qui pourra effectuer toute une série de contrôles et produire des rapports synthétiques à leur sujet.

3.5. LE DIALOGUE UTILISATEUR/SYSTEME

Si l'on considère l'ensemble DMR-XL, l'environnement de travail proposé à l'utilisateur est hybride (XL pour les modèles, DMR pour toutes les autres fonctions). Le langage de commandes proposé par DMR, qui est un produit très technique, est assez lourd risquant ainsi de décourager les utilisateurs; nous pensons surtout aux commandes à utiliser pour créer des relations et à la manipulation d'un member-name non significatif (vu plutôt comme une "database-key"). A ce sujet, il serait d'ailleurs intéressant d'étudier dans quelle mesure les

alias (standardisés) pourraient servir d'identifiant. Ceci est techniquement possible via des contrôles d'unicité d'un groupe d'alias à l'introduction de chaque membre.

XL par contre propose des mécanismes plus intéressants qui allègent considérablement les opérations d'introduction et de mise à jour. Malheureusement, les services que XL propose sont beaucoup plus limités que ceux de DMR surtout au niveau des contrôles et des fonctions d'interrogation.

Dans ces conditions, il serait préférable de considérer DMR comme un outil central d'un système intégré où différents utilitaires (dont XL) prendraient le relais pour l'exécution des opérations demandées par l'utilisateur via une communication homogène. L'interface homme/machine ressemblerait alors à celle de XL mais elle proposerait dans un langage convivial toute une série de fonctions supplémentaires supportées par DMR (interrogation, help, contrôles, édition de rapports, génération de DDL, commandes de mise à jour automatique à partir d'autres systèmes, ...).

On pourrait également prévoir un utilitaire TUTORIAL dont la fonction serait de guider l'utilisateur qui veut documenter son projet suivant la méthodologie.

4. REFLEXIONS CRITIQUES

L'objectif que nous avons assigné au SAD est double. Nous avons montré dans ce chapitre que l'aspect administration des données est théoriquement faisable et que l'aspect d'aide au développement doit être envisagé de manière prudente.

Dans ce paragraphe, nous ferons état de quelques considérations dont il est souhaitable de tenir compte lorsque l'on décide d'installer un outil de ce type.

Tout outil de documentation ou d'aide au développement doit disposer d'informations respectant les caractéristiques suivantes :

- elles doivent être correctes, fidèles au réel perçu,
- elles doivent être centralisées (si deux données se trouvent à des endroits différents, elles doivent être les mêmes; par ailleurs, si l'on dispose de versions multiples d'une même donnée, leur gestion doit être assurée),
- elles doivent être structurées dans la mesure où l'on doit pouvoir les consulter aisément,
- elles doivent être partiellement standardisées pour être compréhensibles et communicables (la standardisation n'exclut cependant pas le texte libre),
- il doit être possible d'établir des relations entre ces informations.

Dans ce cadre, l'utilisation d'un système automatisé se justifie amplement. Mais, dans la réalité, plusieurs problèmes se posent. Nous les examinerons successivement ci-dessous au travers des propriétés auxquelles les informations doivent répondre.

4.1. INFORMATIONS CENTRALISEES

Cette caractéristique implique que toutes les personnes responsables d'éléments constitutifs de la documentation les intègrent dans une base de données centralisée. Or, ceci ne sera possible que moyennant le bon vouloir de ces différentes personnes car cette tâche est assez fastidieuse. Il est donc nécessaire de trouver des "arguments" pour les convaincre de l'utilité d'un dictionnaire des données en prouvant que ce dernier peut offrir un "feed-back" intéressant à ses utilisateurs. Or, nous verrons en 4.2. que les avantages d'un DD sont souvent limités dans la pratique.

Une autre solution à ce problème consiste à charger automatiquement le dictionnaire des données à partir d'autres sources d'informations existant sous forme automatisée. Cette solution est satisfaisante étant donné

que c'est le DD lui-même qui charge sa base de données. Elle est d'autant plus efficace lorsque l'on s'attache à des données opérationnelles (ou de production) car, d'une part, elles existent toujours dans une source quelconque d'informations et, d'autre part, elles sont à jour.

Une fois les données chargées, il faudra procéder à un certain nombre de contrôles de cohérence de ces dernières par rapport aux autres informations du dictionnaire. Remarquons que ces contrôles sont plus aisés dans le cas de données totalement standardisées.

4.2. INFORMATIONS STRUCTUREES ET STANDARDISEES

Notre étude concernant la méthodologie nous a permis de soulever quelques limites dues à l'utilisation de modèles standardisés et à l'utilisation d'une méthodologie unique pour tous les projets.

Ces limites se répercutent immédiatement lorsque l'on veut standardiser les informations à intégrer dans un DD. Si les problèmes de standardisation sont solubles lorsque l'on a affaire à des systèmes bien déterminés comme des schémas SGBD, la structuration modulaire des programme, etc, ils deviennent cruciaux lorsque l'on envisage de standardiser la documentation relative aux phases du développement. Dans ce dernier cas, la recherche du meilleur mode de représentation du réel perçu et de l'idée que l'on a d'un système fait appel, pour chaque cas particulier, au raisonnement et à la créativité.

Partant de ce principe, deux solutions sont envisageables.

La première consiste à garder un outil où les informations et les relations établies entre ces informations sont rigoureusement typées mais de manière assez générale pour pouvoir identifier tous les éléments du SI. L'inconvénient de ceci est que les informations que l'on tirerait d'un tel système risquent d'être assez pauvres parce que beaucoup trop générales.

La seconde optique consiste, à l'autre extrême, à fournir un système que chaque utilisateur définirait selon sa propre vue. Cette solution suppose que le système dispose d'un mécanisme très puissant, pour ne pas dire presque "intelligent", qui puisse mettre en relation toutes les informations du SI et fournir à l'utilisateur un "langage d'interrogation" lui permettant d'avoir une réponse à tout type de question. Cependant, la complexité et le non-réalisme de cette solution nous obligent à renoncer à l'espoir de voir naître un jour un système de ce type.

Néanmoins, il serait intéressant d'étudier dans quelle mesure un système contenant des informations structurées pourrait répondre au besoin de non-standardisation de l'utilisateur car, selon nous, l'un n'exclut pas l'autre. En effet, on pourrait imaginer que la standardisation ne porte que sur les points saillants de la documentation et que l'utilisateur soit libre de documenter à sa manière les informations auxiliaires s'articulant autour de ces points.
Dès lors, le rôle de la méthodologie consisterait à définir quels sont ces points saillants, communs (si possible) à tous les projets.

4.3. INFORMATIONS CORRECTES

Par informations correctes, nous entendons des informations cohérentes et à jour. Cette contrainte est fondamentale. En effet, si la documentation n'est pas à jour, elle perd toute sa crédibilité et est donc moins utilisée. Même si l'on dispose d'un outil aux capacités illimitées, il ne sera pas exploité si les utilisateurs ne s'impliquent pas dans sa tenue à jour. Le problème est donc plus délicat que pour la réticence des utilisateurs à encoder de nouvelles descriptions car, dans le cas présent, le DD pourrait donner une fausse image de la réalité.

Apparemment, le seul moyen de sortir de cette impasse est de proposer des contrôles très stricts. Ils peuvent se traduire par une mise à jour automatique et systématique de certaines données via des interfaces; mais encore une fois, ceci n'est pas applicable pour les phases de la méthodologie où les concepts manipulés ne sont pas toujours formalisables. Dans ce dernier cas, nous devrons donc faire appel à la déontologie...

TROISIEME PARTIE

CONCLUSIONS GENERALES

Au cours de notre étude, nous avons abordé deux aspects : l'aspect méthodologie et l'aspect des outils d'aide à la documentation et au développement de projets.

En ce qui concerne la méthodologie, nous avons été confrontés à un éternel dilemme : standardisation versus liberté. D'une part, la standardisation constitue un frein à la créativité, et d'autre part, la liberté laissée lors du développement de projets peut entraîner le chaos. Cependant, ces deux tendances sont nécessaires pour assurer une meilleure productivité d'autant plus que chaque application a ses caractéristiques propres.

Rappelons les objectifs attribués en général aux méthodologies :

1. développer des logiciels de meilleure qualité;
2. augmenter la productivité en laissant une trace de la démarche adoptée aussi bien pour l'aspect maintenance que pour l'aspect division du travail.

Dans le cas d'une "méthodologie libre", le premier objectif sera atteint si le développeur travaille d'une manière logique et consciencieuse et, en ce qui concerne le deuxième objectif, la documentation sera claire et précise. On peut cependant se demander si une documentation plus standardisée ne serait pas plus appropriée.

Dans le cas d'une méthodologie stricte, il n'est pas évident que l'on puisse obtenir des produits de meilleure qualité car le développeur pourrait alors être obligé d'adopter une démarche qui ne serait pas nécessairement appropriée à son problème. Dans ce cas, on pourrait envisager d'établir de nouvelles règles selon les besoins, ce qui, à long terme, risquerait de rendre la méthodologie incompréhensible. La documentation, quant à elle, reflètera ce que la méthodologie impose et sera alors confrontée au même type de problème.

A partir de cela, on peut conclure que la démarche libre serait la plus avantageuse mais dans la réalité cette démarche n'est pas envisageable et ce, pour deux raisons :

1. Le développeur n'est pas seul face à son projet. Une marche à suivre homogène peut faciliter la communication entre les différentes personnes concernées par le développement

2. La solution de facilité qui consiste à concevoir un système sans une analyse approfondie risque d'être adoptée dans le cas où les échéances sont trop brèves.

Il semble donc nécessaire d'imposer des standards et des contrôles pour être certain qu'une démarche méthodologique soit suivie.

Nous pensons donc qu'une bonne méthodologie devrait imposer une standardisation en vue de limiter le chaos tout en proposant un éventail de techniques en vue de faciliter la tâche du développeur qui devra alors faire preuve de discernement en choisissant les techniques les plus appropriées à son problème. Mais, il ne nous est pas envisageable de définir avec précision le contenu de cette méthodologie.

Dans l'état actuel des choses, il ne nous paraît pas possible d'affirmer que l'application d'une méthodologie apportera un gain du point de vue temps de développement et de maintenance. Il nous semble que des objectifs raisonnables sont la qualité des services rendus aux utilisateurs et une meilleure maîtrise du système développé.

En ce qui concerne les outils, nous avons examiné dans quelle mesure un système automatisé de documentation, aussi puissant soit-il, peut servir de support au développement et de source centralisée d'information au sujet d'un ensemble de projets. Cette source d'information doit pouvoir servir de base à la fonction d'administration des données. Le type de documentation que nous considérons ici concerne non seulement des descriptions d'éléments opérationnels du projet (fichiers, programmes, enchaînements des modules,...) mais aussi des informations dégagées au fil des étapes du développement de projets.

Si l'on considère les éléments opérationnels, leur structure, en général bien définie, rend possible la détermination de standards de documentation.
Le chargement d'un outil de documentation peut se faire de manière manuelle mais, dans ce cas, il peut aussi se faire de manière automatique (à partir de directories d'un SGBD, par exemple). De plus, si une interface dynamique est prévue entre le système de documentation et d'autres outils, la cohérence des informations pourra être assurée à tout moment. Notons qu'une telle interface paraît indispensable si l'on veut pouvoir tirer des informations correctes (c'est-à-dire à jour) d'un outil de documentation.

Selon nous, le principal problème à ce niveau consiste à identifier ces informations de manière précise. En effet, dans tout S.I., il peut exister des informations différentes ayant le même nom ou encore une même information ayant des noms différents, ce qui peut prêter à confusion. Ce problème sera résolu par l'instauration de "naming conventions" dont la portée s'étend à tous les projets.

Si l'on considère les éléments non opérationnels dégagés en amont dans la démarche de développement, quelques problèmes peuvent se poser.

Premièrement, certaines de ces informations ne sont pas formalisables, de ce fait, il faudra quelquefois les exprimer en texte libre. Il est possible de prévoir des systèmes complexes de traitement du texte libre mais, en général, un outil ne pourra pas exploiter ce genre d'informations de manière fiable, de plus, les traitements qu'il offrira resteront souvent limités.

Deuxièment, nous pensons qu'il n'est pas souhaitable de documenter tous les projets de la même manière. Comme nous l'avons mis en évidence dans l'étude d'une méthodologie, chaque projet ou classe de projets a ses caractéristiques propres. Si malgré tout, un système strict de documentation uniforme est imposé, les développeurs seront obligés de transformer leurs analyses pour les rendre conformes au système de documentation, ce qui aura pour conséquence d'y introduire des informations qui ne traduiront pas toujours exactement l'idée du développeur.
Tout système automatisé de documentation nécessite une standardisation assez poussée, nous pensons que cela n'est pas faisable de manière générale. Nous suggérons donc de la déterminer non pas au niveau de tous les projets mais plutôt au niveau plus particulier de chaque classe de projets (par exemple pour les applications développées à l'aide d'APE et DB2).
Selon nous, c'est seulement dans cette optique qu'il sera possible d'exploiter un outil de telle sorte qu'il fournisse de bons résultats.

Un troisième problème souvent propre aux éléments non opérationnels est celui de l'adéquation de leur description avec la réalité. En effet, la mise à jour automatique d'un système de documentation n'est envisageable que pour des données étant enregistrées sous forme automatisée, ce qui est rarement le cas des données non opérationnelles. Leur mise à jour devra donc se faire de manière manuelle. Dans ce cadre, il faudra trouver les moyens de convaincre les responsables d'un projet de le documenter. A ce niveau, si l'on ne peut s'appuyer sur la bonne volonté, il faudra recourir à des contrôles étroits du processus de documentation pour tous les projets, ce qui est délicat à instaurer dans toute organisation.

Mais, si un outil de documentation comporte des risques de contenir des informations incorrectes, on pourrait se demander s'il joue encore un rôle de documentation où si au contraire, il ne risque pas d'induire ses utilisateurs en erreur. Nous pensons malgré tout qu'il vaut mieux disposer d'un système même imparfait plutôt que de n'avoir aucun système de documentation du tout. En effet, un tel système permet à l'organisation de s'interroger sur la structure de son S.I. et de dégager les relations existant entre ses différents composants. Même si toutes ces informtions ne sont pas toujours tenues à jour, elles permettent une meilleure maîtrise de ces données ce qui constitue un atout non négligeable.

BIBLIOGRAPHIE

ABIS : A guided tour of Excelerator, Index Technology Corporation, (Massachusetts), 1984, 67 p.

ALLAN, Frank W., LOOMIS, Mary E.S., MANNINO, Michael V. : The integrated Dictionary/Directory System, in : ACM Computing Surveys, vol. 14, n° 2, juin 1982, pp. 245-280.

BODART (a), François, PIGNEUR, Yves : Conception assistée des applications informatiques, 1. étude d'opportunité et analyse conceptuelle, (MASSON, Presses Universitaires de Namur), 1983, 241 p.

BODART (b), François, TEICHROEW, Daniel : Les outils d'aide à la conception d'un système d'information - Le projet ISDOS, in : Informatique et Gestion, n°125, juin 1981, pp. 47-55.

CANNING, Richard G. : Installing a Data Dictionary, in : E.D.P.-Analyser, vol 16, n° 1, janvier 1978, pp. 1-12.

DIGNUM, F., JACQUES, M., KERSTEN, M.L. : Workbenches : een ontwerpomgeving voor informatiesystemen, in : Informatie, n° 28, octobre 1986, pp. 793-896.

DOLK, Daniel. R., KIRSCH, Robert A. : A relational information resource dictionary system, in : Communication of the ACM, vol. 30, n° 1, janvier 1987, pp. 48-61.

FRENKEL, Karen A. : Toward automating the software-development cycle, in : Communication of the ACM, vol. 26, n°6, juin 1985,

GRAAS (a), Christian : Introduction à la méthodologie, cours MAI, (Caisse Générale d'Epargne et de Retraite, Bruxelles), juin 1985, 72 p.

GRAAS (b), Christian : Définition de projet, (Caisse Générale d'Epargne et de Retraite, Bruxelles), juin 1986, 18 p.

GRAAS (c), Christian : Cours d'introduction à IMS, (Caisse Générale d'Epargne et de Retraite, Bruxelles), 1986.

HAINAUT (a), Jean-Luc : Synthèse de la table ronde sur l'administration des données, (Namur), juillet 1987, 12 p.

HAINAUT (b), Jean-Luc : Conception assistée des applications informatiques, 2. conception de la base de données, (MASSON, Presses Universitaires de Namur), 1986, 206 p.

JANSSEN, T.G.M., GERSTELING, H., PEETERS, J.M.P. : Data Dictionary/Directory, in : Informatie, n° 20, octobre 1978, pp. 576-649.

KERSCHBERG, Larry, NAVATHE, Shamkant : Role of Data Dictionaries in information resource management, in : Information & Management, n° 10, 1986, pp. 21-46.

LUYCKX, Willy : Manuel pour l'utilisateur du dictionnaire des données DATAMANAGER, (Caisse Générale d'Epargne et de Retraite, Bruxelles).

MAI (a) : Analyse conceptuelle des données, version 1.0, (Caisse Générale d'Epargne et de Retraite, Bruxelles), 22p.

MAI (b) : Aperçu de la filière de développement de projets informatiques, version 1.0, (Caisse Générale d'Epargne et de Retraite, Bruxelles), 5 p.

MAI (c) : Demande de développement, version 0.4, (Caisse Générale d'Epargne et de Retraite, Bruxelles), 7 p.

MAI (d) : Développement de projets "IMS", (Caisse Générale d'Epargne et de Retraite, Bruxelles), 9 p.

MAI (e) : La normalisation, version 1.0, (Caisse Générale d'Epargne et de Retraite, Bruxelles), 19 p.

MAI (f) : Méthodologie et développement de projets, chapitre 3: analyse conceptuelle, version 0.3, (Caisse Générale d'Epargne et de Retraite, Bruxelles), 21 p.

PEETERS, Emile : Systèmes d'analyse de l'information et banques d'information, fascicule 1, (Presses Universitaires de Bruxelles), 1979-1980, pp. 52-82.

WALTER, Jacques : Administration des données, formation CAP SOGETI, Montparnasse Park Hotel, janvier 1985.

A 1 a

C.G.E.R.

C.T.I.

DEMANDE DE DEVELOPPEMENT

version 0.4
du 26/08/86

A1b

AVANT-PROPOS

Pourquoi ce formulaire?

Pour venir en aide aux utilisateurs de l'informatique en permettant au CTI d'apprécier plus aisément les demandes qui lui sont soumises.

Aussi ce dernier se tient-il à la disposition de ces utilisateurs pour remplir avec eux ce formulaire.

Quand le remplir?

Lors d'une demande
- d'un nouveau système, ou
- d'une modification ou d'une extension conceptuelle d'un système existant
  (par exemple: nouvelle procédure à automatiser, nouveau fichier, nouvelle donnée, etc...),
- d'un nouveau matériel.

Dans les autres cas, on continuera d'utiliser le formulaire mod M D.P. 6501 (par exemple, pour une demande d'extension de zone, ou de changement d'une règle de controle ou de calcul, etc...).

Comment remplir ce formulaire?

Au verso des pages suivantes vous trouverez, en regard de chaque rubrique, des indications destinées à en faciliter la rédaction.

Si vous estimez ne pas avoir à en remplir l'une ou l'autre, veuillez en indiquer la raison (pas applicable, référence vers une autre rubrique, inconnu, etc...).

Si, au contraire, vous ne disposez pas d'assez de place, renvoyez à une ou plusieurs annexes.

Enfin, n'hésitez pas à nous faire part de toutes les remarques et suggestions susceptibles d'améliorer ce formulaire.

MERCI D'AVANCE.

A1c

C.G.E.R.		p.3
	DEMANDE DE DEVELOPPEMENT	
C.T.I.		version 0.4

1. GENERALITES.

DATE DE REDACTION :

INTITULE DU DEVELOPPEMENT :

SERVICE(S) CONCERNE(S) : No(s)

FONCTIONNAIRE RESPONSABLE DE LA DEMANDE : NOM : TEL :

SIGNATURE :

MANAGER UTILISATEUR (Rang 13 minimum) : NOM : TEL :

SIGNATURE :

CODE(S) PPBS DU(DES) SERVICE(S) UTILISATEUR(S) ET REPARTITION DES COUTS

Code	%	Service

LISTE DES MEMBRES DU GROUPE DE TRAVAIL

2. DESCRIPTION DU CONTEXTE ACTUEL.

SITUATION ACTUELLE :

Décrivez-la succinctement, en mentionnant en outre, s'il y a lieu, les procédures administratives, les obligations légales, les conventions avec des organismes extérieurs à la CGER, les liens avec d'autres applications informatiques, etc... .

PRODUIT(S) CONCERNE(S) :

Produits et gestions touchés par le système.

ESTIMATION DES PRINCIPAUX VOLUMES D'INFORMATION A TRAITER

On indiquera ici, notamment, le nombre de contrats, de personnes, de comptes, etc.. gérés, le nombre de de transactions introduites quotidiennement, etc... .

3. CAUSES ET ORIGINE DU PROBLEME.

Mentionnez par exemple les problèmes concernant les coûts, la fiabilité, les temps de réponse, les modifications éventuelles de certaines contraintes légales ou administratives, etc... .

2. DESCRIPTION DU CONTEXTE ACTUEL.

SITUATION ACTUELLE :

PRODUIT(S) CONCERNE(S) :

ESTIMATION DES PRINCIPAUX VOLUMES D'INFORMATION A TRAITER

3. CAUSES ET ORIGINE DU PROBLEME.

4. AVANTAGES ATTENDUS

4.1. AVANTAGES QUANTIFIABLES.

Chiffrez, en francs, la rentabilité, les économies de personnel, etc....

4.2. AVANTAGES NON QUANTIFIABLES.

On mentionnera ici, par exemple l'utilité stratégique, l'amélioration des procédures administratives, ou celle du du service rendu à la clientèle, etc... .

5. CONTRAINTES DE REALISATION.

5.1. PREVOYEZ-VOUS DIFFERENTES PHASES DE DEVELOPPEMENT?

Si un développement en phases est possible, mentionnez-les ici, ainsi que les dates de réalisation souhaitées pour chacune.

5.2. BUDGET DE DEVELOPPEMENT ENVISAGE.

4. AVANTAGES ATTENDUS

4.1. AVANTAGES QUANTIFIABLES.

4.2. AVANTAGES NON QUANTIFIABLES.

5. CONTRAINTES DE REALISATION.

5.1. PREVOYEZ-VOUS DIFFERENTES PHASES DE DEVELOPPEMENT?

5.2. BUDGET DE DEVELOPPEMENT ENVISAGE.

PPLICATION CODE ASSIGNMENT FOR

PLEASE SEND BACK TO SECURITY ADMINISTRATOR (61.6)

TYPE OF REQUEST: CREATE (1) UPDATE (1) DELETE (1)

TO BE COMPLETED BY CTI-IVC.

APPLICATION IDENTIFICATOR (XX) (2): DEVELOPMENT GROUP (3): ACCOUNT NB (2):

APPLICATION DESCRIPTION (2)

DATA DICTIONARY INFORMATION (2): DICTIONARY NAME: DEVELOPMENT STATUS: PROJECT CODE:

DEVELOPMENT PEOPLE	NAME	SURNAME	TEL	USERID
PROJECT LEADER (2):				
BACK-UP (3):				
DOCUMENTALIST (3):				

TO BE COMPLETED BY OWNER

OWNER (USER 'S NAME (2): (RANK > OR = 13) SURNAME: TEL: SERVICE:

BACK UP'S NAME (3): SURNAME: TEL: SERVICE:

CONFIDENTIAL DATA (Y/N) (2):

PPBS-CODE(S) (2):

DEVELOPMENT	DEVELOPMENT CONTROL	DATA ADMINISTRATION	USER(5)
MANAGER SIGN (3):	SIGNATURE (4):	DATA ADM. SIGN (2):	OWNER SIGN (2):
NAME:	NAME:	NAME:	NAME:
DATE:	DATE:	DATE:	DATE:
PROJECT LEADER SIGN (2):		METHODOLOGY SIGN (4):	BACK-UP-SIGN (3):
NAME:		NAME:	NAME:
DATE:		DATE:	DATE:

(1) CIRCLE WHEN APPLICABLE (2) MANDATORY (3) OPTIONAL (4) IN CASE OF CREATE/DELETE ONLY
(5) IN CASE OF CHANGE OF OWNER, THE PREVIOUS OWNER MUST SIGN

A2a

LEGENDE DE LA DEMANDE DU CODE D'APPLICATION

La demande du code d'application se décompose en trois parties :

- la première partie doit être complétée par le CTI,
- la deuxième partie doit être complétée par l'utilisateur,
- la troisième partie est réservée aux signatures pour approbation; à ce stade, il n'y a pas d'ordre précis pour l'obtention des signatures mais le document doit toujours parvenir en dernier lieu au service sécurité pour accorder les priorités nécessaires.

Dans la première partie :

- back up = correspondant habituel,
- project leader = chef de projet.

Dans la deuxième partie :

- owner = sous-directeur.

Dans la troisième partie :

1. Développement.
2. Development control pour la signature du responsable du passage en production qui attribue un code d'application.
3. Data administration : passage au MAI où l'on complète les données dictionnaire (ddict).
4. Utilisateur.

DIAGRAMME GENERAL TRAITEMENT-UTILISATEUR

REF AC/TK : référence de l'activité ou de la tâche
ou si la place le permet, nom sous lequel le traitement est connu

TEMPS : J, J+1 ...

DUREE : préciser l'unité

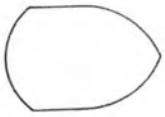
REF. RESPONSABLE : référence ou nom du responsable de l'exécution du traitement). Si on utilise des références (p.ex. US....) ne pas oublier d'en indiquer la signification en annexe (p.ex. US 103 : CTI)

TYPE . B = Batch
I = Interactif
M = Manuel

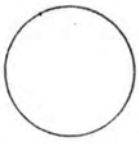
REF AC/TK	TEMPS	DUREE	TYPE	REF. RESPONSABLE				
1. 2. 3. : :								

DIAGRAMME DETAILLE TRAITEMENT - UTILISATEUR

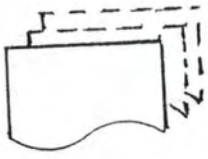
Symboles utilisés :



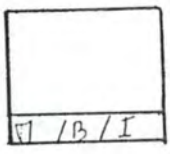
écran



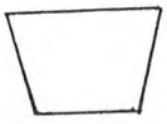
fichier magnétique



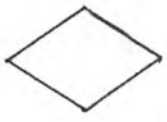
documents / listes / microfilm(s)
(... : si plusieurs copies)



traitement
M = Manuel B = Batch I = Interactif



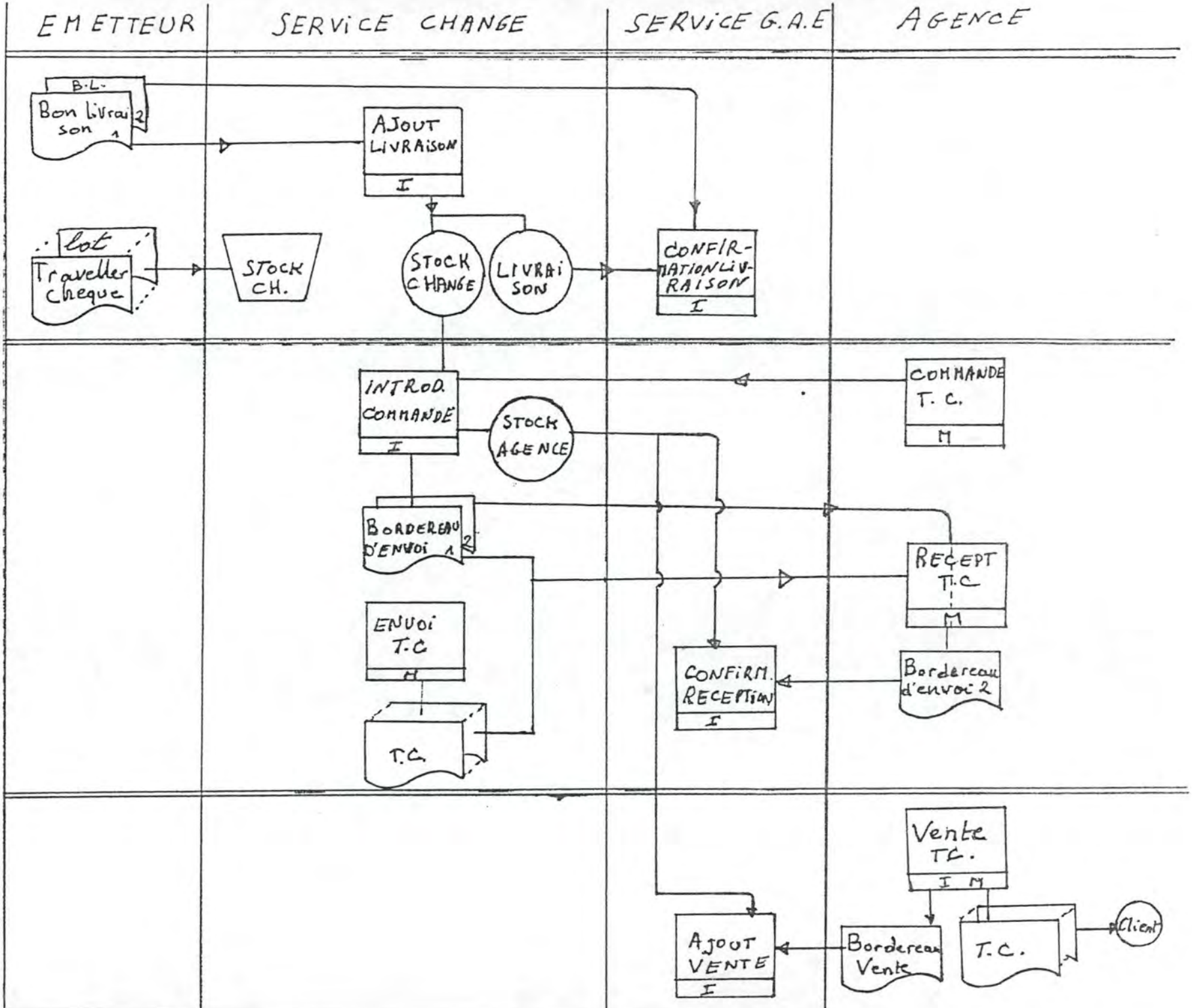
objet physique (?)



décision

connecteur

A4 b


EMETTEUR
SERVICE CHANGE
SERVICE G.A.E
AGENCE
B.L.
Bon livraison 1 2
AJOUT LIVRAISON
I
lot
Traveller Cheque
STOCK CH.
STOCK CHANGE
LIVRAISON
CONFIRMATION LIVRAISON
I
INTROD. COMMANDE
I
STOCK AGENCE
COMMANDE T. C.
M
BORDEREAU D'ENVOI 1 2
ENVOI T.C
M
T.C.
CONFIRM. RECEPTION
I
RECEPT T.C
M
Bordereau d'envoi 2
Vente TC.
I M
AJOUT VENTE
I
Bordereau Vente
T.C.
Client

1. GERER LES RELATIONS EXTERNES

1.1. GERER LES SOCIETES

1.1.1. CREATION SOCIETE

1.1.2. MAJ SOCIETE

1.1.3. EXPEDIER PROJET

1.1.4. SUPPRIMER SOCIETE

1.1.5. INITIALISER UN GROUPE

1.2. GERER LES GROUPES

1.2.1. ASPECTS SIGNALETIQUES GROUPES

1.2.1.1. MAJ DONNEES SIGNAL.D'UN GROUPE

1.2.1.2. SUPPRIMER UN GROUPE

1.2.2. REGLES DU GROUPE

1.2.2.1. REDIGER CONVENTION ET REGLEMENT

1.2.2.2. CREATION NOUVEAU REGLEMENT

1.2.2.3. CREATION REGLES D'INTRODUCTION

1.2.2.4. ADAPTATION DU REGLEMENT EN VIGUEUR

1.2.2.5. ADAPTATION REGLES D'INTRODUCTION

1.2.3. LIAISONS D'UN GROUPE

1.2.3.1. MAJ LIAISON GRP/ PROD.

1.2.3.2. MAJ LIAISON GRP/SOC.

1.2.3.3. MAJ LIAISON GRP/ F.S.

1.2.3.4. MAJ LIAISONS GRP/ P.B.

1.2.3.5. MAJ LIAISON GRP/INDIVIDU

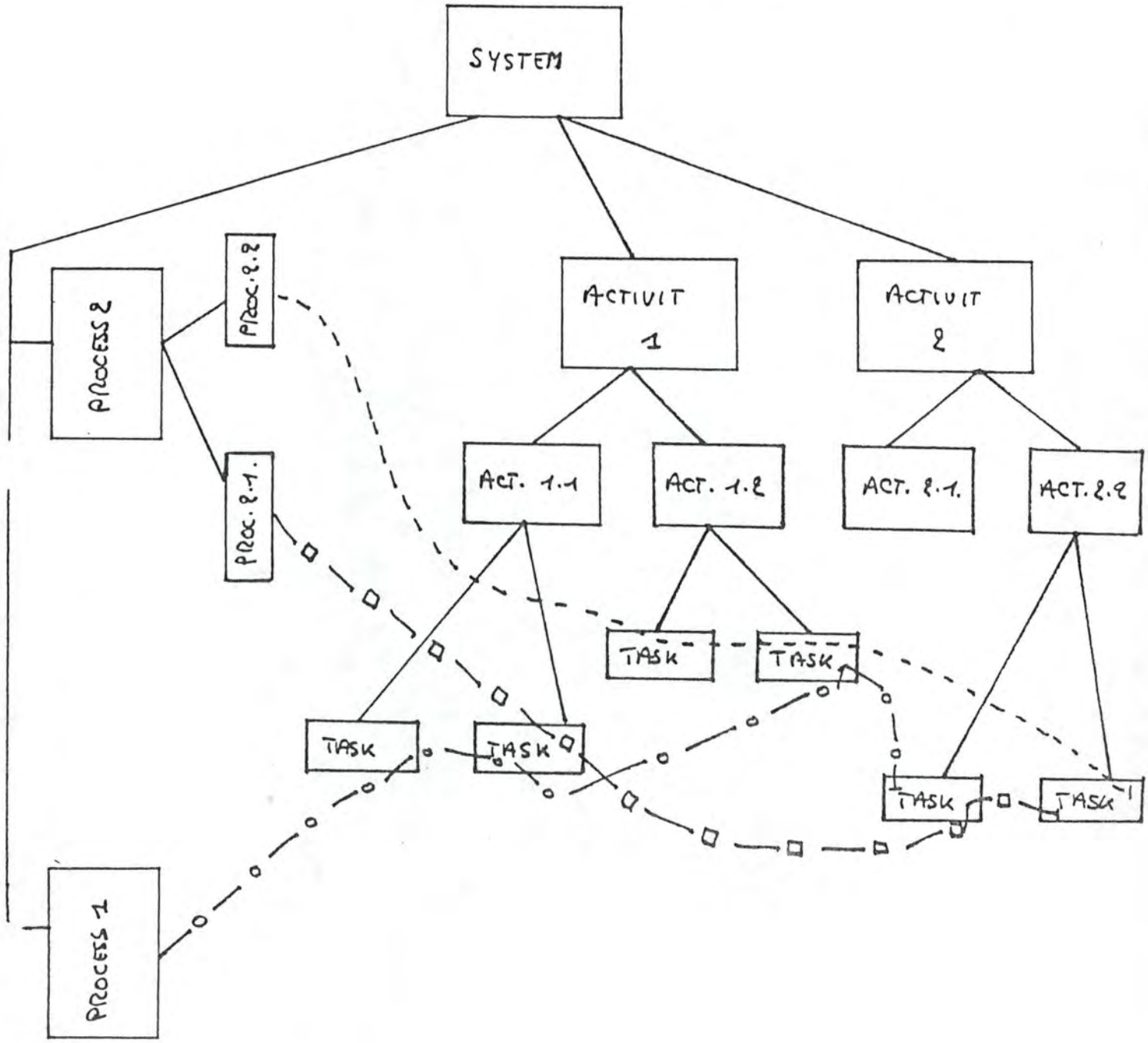
1.2.4. GESTION DE L'ENTITE GROUPE

1.2.4.1. RECEPTION & CTRL LISTE SAL/PRIMES

1.2.4.2. ENREGISTRER SAL/PRIME PAR VIDEO

1.2.4.3. ENREGISTRER SAL/PRIME PAR FICHIER

MODELE DE LA DYNAMIQUE



METHODOLOGIE

DIAGRAMME TACHES - ENTITES / TAKEN - ENTITEITEN SCHEMA

I = INPUT
O = OUTPUT
U = UPDATE
C = I + O

ENTITES
ENTITEITEN

TACHES
TAKEN

TKM	DESCRIPTION BESCRIJVING	SEQ	FOURNISSEURS	NOM FOURNISSEUR	ORDRES ACHAT	ARTICLES	COMMANDES	LIGNE COMMANDE	BONS ACHAT	CLIENTS	NOM CLIENT	ARCH. FOURNISSEUR	ARCHIVES ARTICLES	ARCH. BONS ACHAT	ARCHIVES COMMANDES
501	AJOUTER UN ARTICLE		I	I	O	O									
502	RECEPTIONNER LES MARCHANDISES DU FOURNISSEUR		I	I		U	U	U	U						
503	MODIFIER CARACTERISTIQUES ARTICLE					U									
504	SUPPRIMER UN ARTICLE					U									
505	ARCHIVER LES ARTICLES					U			I				O		
506	ETABLIR ET ENVOYER LES BONS D'ACHAT				I	I			O						
507	ARCHIVER LES ACHATS								U					O	
508	AJOUTER UN FOURNISSEUR		O	O											
509	MODIFIER CARACTERISTIQUES FOURNISSEUR		U	U											
510	SUPPRIMER UN FOURNISSEUR		U	U	I				I						
511	ARCHIVER FOURNISSEURS		U	U					I			O			
512	AJOUTER UN CLIENT									O	O				
513	MODIFIER LES CARACTERISTIQUES CLIENT									U	U				
514	NORMALISER LES BONS DE COMMANDE									I	I				
515	TRAITER LES BONS DE COMMANDE				O	U	O	O		I	I				
516	CONSULTER COMMANDES EN COURS						I	I	I	I	I				
517	ARCHIVER COMMANDES						U	I							O
518	EXECUTER LES COMMANDES.					I	U	U		I	I				

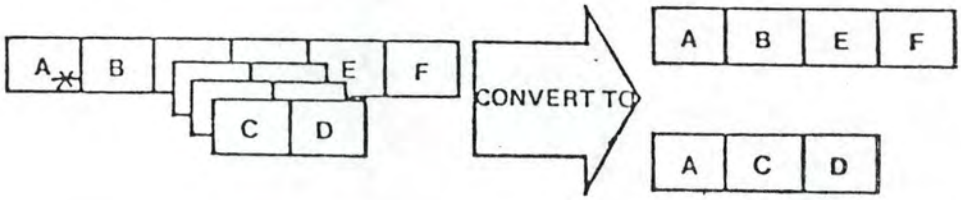
LA NORMALISATION

Notations:

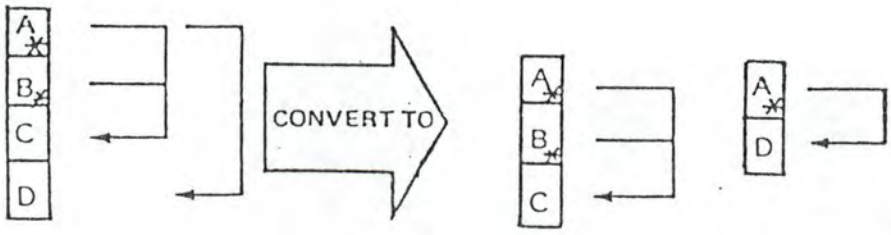
A * ⇒ A est clé primaire

A ⌐
B ↙ ⌐ ⇒ B dépend fonctionnellement de A

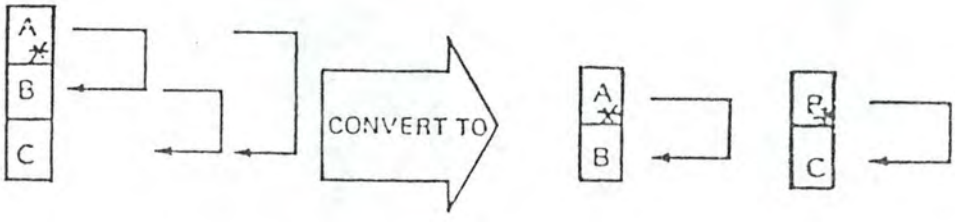
1 NF



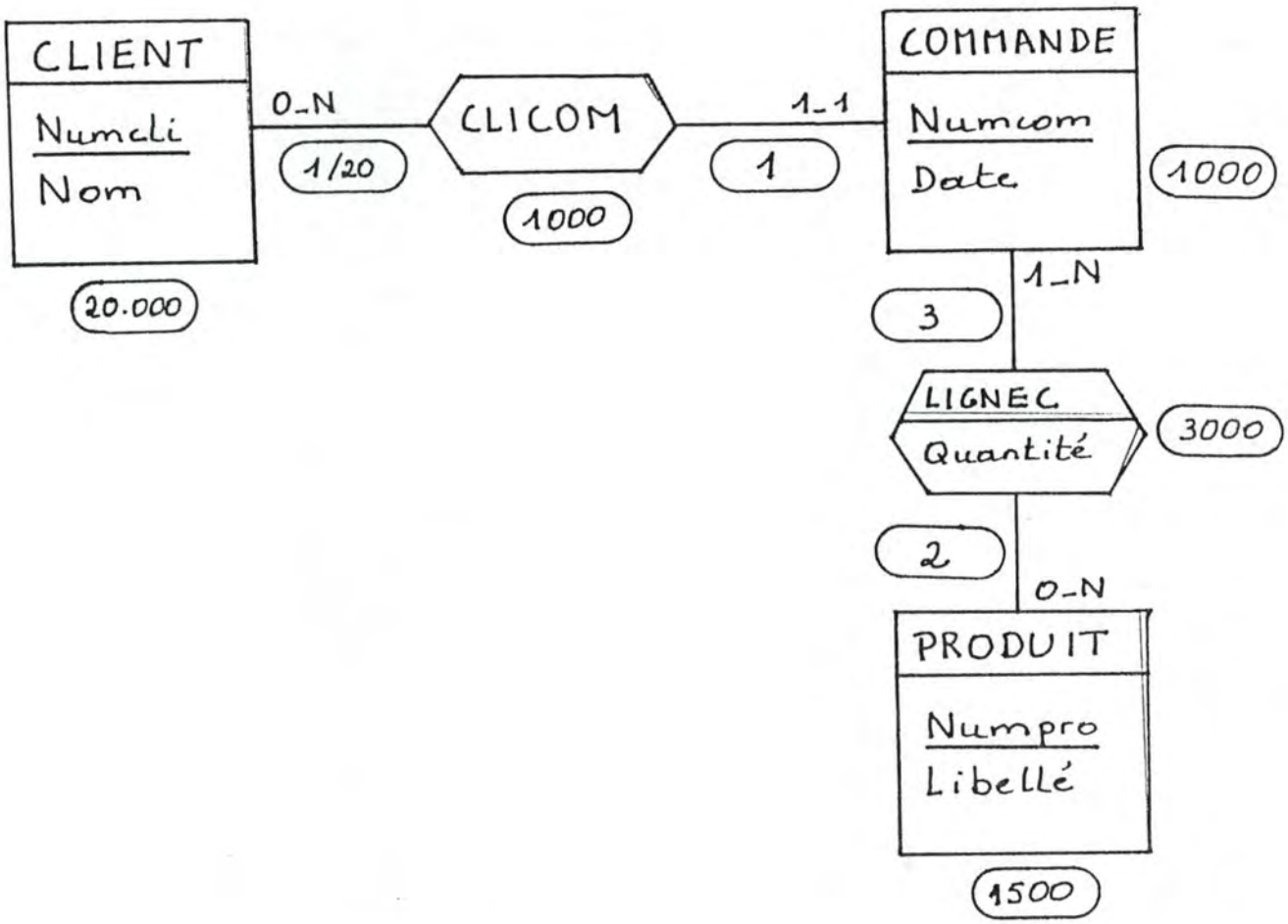
2 NF



3 NF



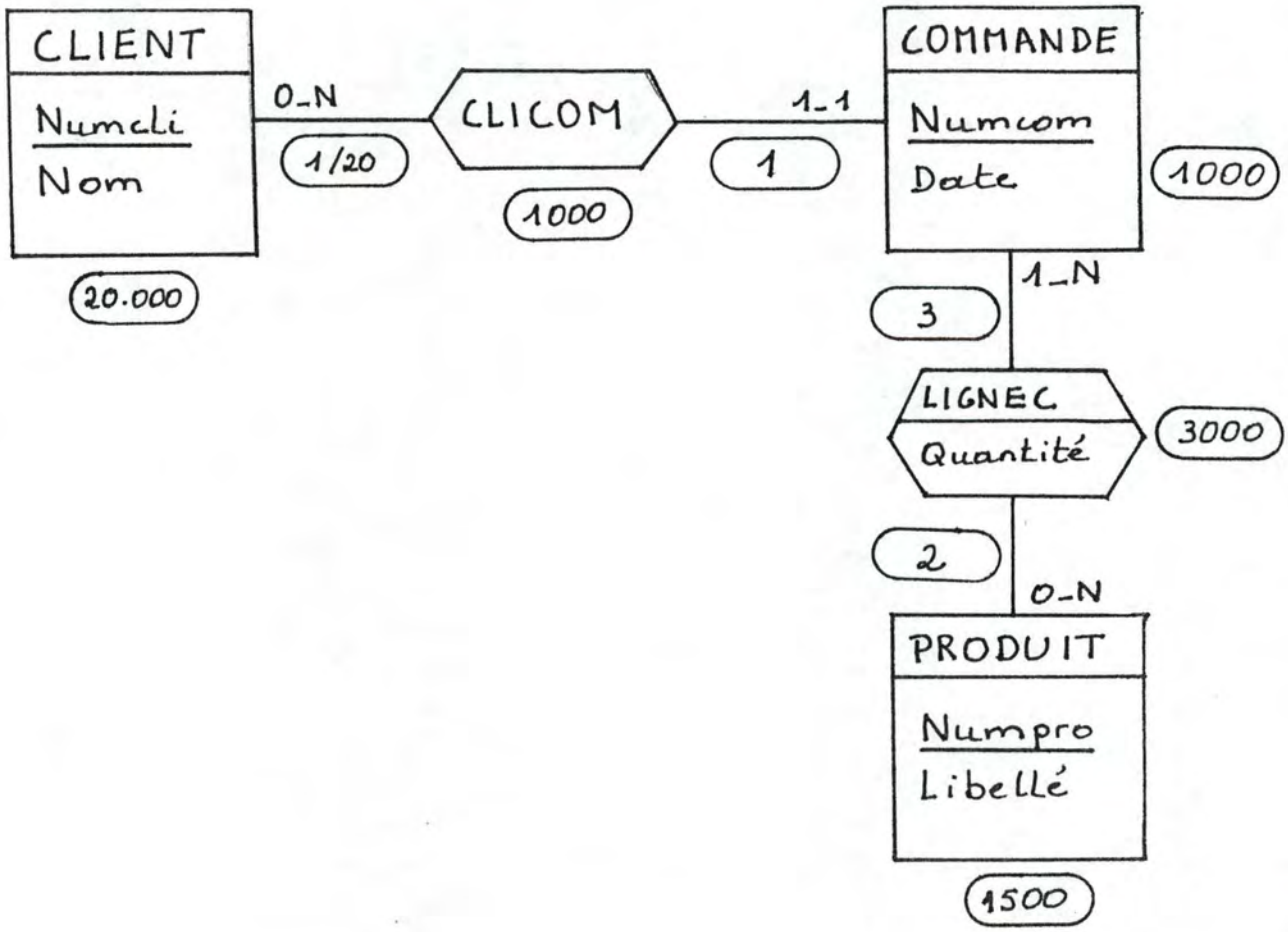
MODELE STATIQUE D'UN SOUS-SCHEMA



( ) contient les quantifications :

- volume des entités, des relations,
- nombre moyen de relations par entité.

MODELE STATIQUE D'UN SOUS-SCHEMA



( ) contient les quantifications :

- volume des entités, des relations,
- nombre moyen de relations par entité.

DATAMANAGER MEMBER DEFINITIONS

Process Members

SYSTEM, PROGRAM, MODULE

SYSTEM, PROGRAM and MODULE members have identical syntax, except for the member type identifier keyword at the start of their definitions:

```
{SYSTEM  }
{PROGRAM }
{MODULE  }

    [CONTAINS member-name [, member-name] ...]

    [INPUTS data-name [, data-name] ...]

    [OUTPUTS data-name [, data-name] ...]

    [UPDATES data-name [, data-name] ...]

    [{CALLS process-name [, process-name]...}]
     {calls-clause-2...                      }

    [{PARAMETERS data-name [, data-name]...     }]
     {entry-point-clause [entry-point-clause]...}

    [LANGUAGE 'string'] [AUTHOR 'text'] [INSTALLATION 'text']

    [DATE-WRITTEN date] [SOURCE-COMPUTER 'text']

    [OBJECT-COMPUTER 'text'] [SPECIAL-NAMES 'text']

    [I-O-CONTROL 'text'] [ASSIGNMENT 'text']

    [EDIT-INPUT 'text'] [EDIT-OUTPUT 'text'] [EDIT-UPDATE 'text']

    [common-clauses]

{;}
{.}
```

where

member-name identifies a member at the same or a lower level in the member type hierarchy

process-name identifies a process member at the same or a lower level in the member type hierarchy

data-name identifies a data member

calls-clause-2 is:

```
    CALLS process-name [AT 'string']
          [PASSING data-name [, data-name]...]
```

entry-point-clause is:

```
    ENTRY-POINT 'string' [PARAMETERS data-name [, data-name]...]
```


Data Members

FILE

```
FILE [{ENTERED-AS }] [{ALIGNED     }]
     {HELD-AS     }  {UNALIGNED   }
     {REPORTED-AS }  {NOT-ALIGNED}
     {DEFAULTED-AS}

    [CONTAINS content [IF clause] [ELSE content [IF clause]]...

        [, content [IF clause] [ELSE content [IF clause]]...]...]

    [{SEQUENTIAL }...] [{NO-LABELS                 }]
     {INDEXED    }     {STANDARD-LABELS            }
     {DIRECT     }     {USER-LABELS module-name    }
     {KEYED      }
     {VSAM       }
     {PARTITIONED}

    [{FIXED [block-size [record-size]]                                  }]
     {VARIABLE [maximum-block-size [maximum-record-size]]               }
     {UNDEFINED [maximum-block-size]                                    }
     {SPANNED [maximum-block-size [maximum-record-size]]                }

    SORT-KEY {item-name } [{ASCENDING }]
             {group-name }  {DESCENDING}

         [, {item-name } [{ASCENDING }]]...]
            {group-name }  {DESCENDING}

    [GENERATION-CYCLE integer]

    [RETENTION-PERIOD integer [{'string'}]]
                              {DAYS    }
                              {MONTHS  }
                              {YEARS   }

    [GROWTH-RATE integer [PERCENT] {'string'}]
                                   {MONTHLY }
                                   {YEARLY  }

    [DEVICE {device [{number  }]}] [DENSITY integer]
            {       {'model' }}
            {number            }
            {'model'           }

    [VOLUME 'name']

    [SIZE size [{KBYTES  }] [, size [{KBYTES  }]]]
               {TRACKS   }           {TRACKS   }
               {CYLINDERS}           {CYLINDERS}
               {'string' }           {'string' }

    [common-clauses]

{;}
{.}
```

where

content is:

```
{file-name                                                  }
{group-name                                                 }
{item-name [version]                                        }
{                                                           }
{{integer       }  {group-name       }                      }
{{item-name-a [version]}  {item-name [version]}             }
```

```
    [KNOWN-AS local-name] [{ALIGNED     }]
                          {UNALIGNED   }
                          {NOT-ALIGNED}
```


clause is:

```
b {EQ }  {c      }  [{AND}  b {EQ }  {c      }]...
  {=  }  {literal}   {OR }    {=  }  {literal}
  {NE }                       {NE }
  {GT }                       {GT }
  {>  }                       {>  }
  {GE }                       {GE }
  {LT }                       {LT }
  {<  }                       {<  }
  {LE }                       {LE }
```

b is item-name-b [version-b]

c is item-name-c [version-c]

GROUP

```
GROUP [{ENTERED-AS }] [{ALIGNED     }]
      {HELD-AS     }  {UNALIGNED   }
      {REPORTED-AS }  {NOT-ALIGNED}
      {DEFAULTED-AS}

    [CONTAINS content [IF clause] [ELSE content [IF clause]]...

        [, content [IF clause] [ELSE content [IF clause]]...]...]

    [KEYS {item-name } [{UNIQUE    }] [{ASCENDING }]
          {group-name }  {DUPLICATED}  {DESCENDING}

        [, {item-name } [{UNIQUE    }] [{ASCENDING }]...]
           {group-name }  {DUPLICATED}  {DESCENDING}

    [USER-EXIT module-name]

    [common-clauses]

{;}
{.}
```

where

content is:

```
{{item-name [version]} [{ALIGNED     }] [KNOWN-AS local-name]                    }
{{group-name         }  {UNALIGNED   }                                           }
{                       {NOT-ALIGNED}                                           }
{{integer       }  {item-name [version]} [{ALIGNED     }]                       }
{{item-name-a [version]}  {group-name  }  {UNALIGNED   }                        }
{                                         {NOT-ALIGNED}                         }
{   [KNOWN-AS local-name] [INDEXED-BY index-name]...                           }
```

clause is:

```
b {EQ }  {c      }  [{AND}  b {EQ }  {c      }]...
  {=  }  {literal}   {OR }    {=  }  {literal}
  {NE }                       {NE }
  {GT }                       {GT }
  {>  }                       {>  }
  {GE }                       {GE }
  {LT }                       {LT }
  {<  }                       {<  }
  {LE }                       {LE }
```

b is item-name-b [version-b]
c is item-name-c [version-c]

B1a

ITEM

```
ITEM [ { [ENTERED-AS ]  [version] }
       { [HELD-AS    ]            }
       { [REPORTED-AS]            }
       {                          }
       { DEFAULTED-AS             }  form-description

       [USAGE { DATE    }]
              { MONEY   }
              { POINTER }

       [CONTENTS content [IF clause] [ELSE content [IF clause]]...]

       [HEADINGS 'string' [, 'string']...]

       [USER-EXIT module-name]]...

    [common-clauses]

{;}
{.}
```

where

form-description is:

```
{ { { [ALPHABETIC  ]  [ [LEFT-JUSTIFIED ] ] }
  { [ALPHAMERIC  ]    [ [RIGHT-JUSTIFIED] ] }
  { [ALPHANUMERIC]                          }
  { [CHARACTER   ]                          }
  {                                         }
  { { [BITS          ] }                    }
  { { [FLOATING-POINT] }                    }
  { { [HEXADECIMAL   ] }                    }
  {                                         }
  { { [BINARY             ]  [ [SIGNED   ] ]  [ [ROUNDED  ] ] } }
  { { [DECIMAL-PACKED     ]  [ [UNSIGNED ] ]  [ [TRUNCATED] ] } }
  { { [PACKED-DECIMAL     ]                                   } }
  { { [NUMERIC-CHARACTER  ]                                   } }
  {                                                           }
  { [ [SIGNED  ] ] [BINARY             ]                      }
  { [ [UNSIGNED] ] [DECIMAL-PACKED     ]                      }
  {                [PACKED-DECIMAL     ]                      }
  {                [NUMERIC-CHARACTER  ]                      }
  {                                                           }
  {   [ [VARIABLE       ] ] [COMPRESSED] [p TO] q             }
  {   [ [NULL-SUPPRESSED] ]                                   }
  {   [ [FIXED          ] ]                                   }
  {                                                           }
  {   [WITH [SEPARATE] { LEADING  } SIGN]                     }
  {                    { TRAILING }                           }
  {                                                           }
  { PICTURE 'picture'                                         }
  {                                                           }
  { NAME item-name-r [version-r]                              }
```

content is:

```
[ { ALPHABETIC        } ] [FORMAT 'format']
  { ALPHAMERIC        }
  { ALPHANUMERIC      }
  { CHARACTER         }
  { BITS              }
  { BINARY            }
  { DECIMAL-PACKED    }
  { PACKED-DECIMAL    }
  { HEXADECIMAL       }
  { NUMERIC-CHARACTER }
  { FLOATING-POINT    }

    [ { [RANGE literal TO literal [, literal TO literal]... ] }
      { [IS literal [TO literal] ], literal [TO literal]]... }

    [CONDITION-NAME condition-name]
```

clause is:

```
b { EQ  } { c       } [ { AND } b { EQ  } { c       } ]...
  { =   } { literal }   { OR  }   { =   } { literal }
  { NE  }               { NE  }
  { GT  }               { GT  }
  { >   }               { >   }
  { GE  }               { GE  }
  { LT  }               { LT  }
  { <   }               { <   }
  { LE  }               { LE  }
```

where

b is item-name-b [version-b]

c is item-name-c [version-c]

picture symbols are:

PICTURE TYPE	PICTURE SYMBOL																								
	A	B	CR	DB	E	I	K	P	R	S	T	V	X	Y	Z	0	9	]	+	-	*	,	.	/
Alphabetic	*																							
Alphanumeric	*												*				*							
Alphanumeric Edited	*	*											*			*	*							
Numeric								*				*					*							
Numeric Edited		*		*		*		*	*	*	*	*		*	*	*	*	*	*	*	*	*	*	*
Numeric Floating Point						*	*	*		*		*	*				*		*	*			*	

* indicates permissible combinations of symbols within picture type

format symbols are:

FORM-DESCRIPTION	FORMAT SYMBOL										
	A	N	O	P	S	X	+	-	.	/	DMY
ALPHABETIC	*				*						
ALPHAMERIC	*	*	*	*	*		*	*	*		*
ALPHANUMERIC	*	*	*	*	*		*	*	*		*
BINARY							*	*	*		*
BITS											
CHARACTER	*	*			*	*				*	*
DECIMAL-PACKED PACKED-DECIMAL		*					*	*	*		*
FLOATING-POINT		*					*	*	*		
HEXADECIMAL											
NUMERIC-CHARACTER		*	*	*			*	*	*		*
PICTURE	*	*	*	*	*	*	*	*	*	*	*

* indicates permissible combinations of symbols and key words

COMMON CLAUSES

B1b

Common clauses apply to all MANAGER Product member types. Thus, for DATAMANAGER they apply to the basic set (SYSTEM through ITEM); to COMMAND-STREAM and OBSOLETE-DEFINITION; to database-specific sets (ADABAS, DL/I, IMS, IDMS, Siemens UDS, SYSTEM 2000/80, TOTAL); to the MARK IV set; and to User Defined sets (EDPS, SAS, SDS, user). Keyword truncation limits depend on the other keywords available in the data definition in which they appear; the limits shown below are unambiguous in any DATAMANAGER basic member definition. Further truncation may be acceptable if the truncated form remains unambiguous for the particular member type.

common-clauses is any of the following clauses:

```
ACCESS-AUTHORITY 'name' [ { READ-ONLY      } ]
                         { UPDATE        }
                         { SECURITY-CONTROL }

    [, 'name' [ { READ-ONLY      } ]]...
                { UPDATE        }
                { SECURITY-CONTROL }

ADMINISTRATIVE-DATA 'text'

ALIAS [alias-type] 'alias' [, [alias-type] 'alias']...

CATALOGUE 'classification' [, 'classification']...

COMMENT 'text'

DESCRIPTION 'text'

EFFECTIVE-DATE date

FREQUENCY { frequency                                                          }
          { { ACCESS } frequency [ { ACCESS } frequency]...                    }
          { { RUN    }             { RUN    }                                 }
          { { UPDATE }             { UPDATE }                                 }
          { { BACK-UP }            { BACK-UP }                                }
```

where frequency is:

```
{ { MONTHLY } [ { number  } ] }
{ { WEEKLY  }   { 'string' } }
{ { DAILY   } }
{ { HOURLY  } }
{ number      }
{ 'string'    }

NOTE 'text'

OBSOLETE-DATE date

QUERY 'text'

SECURITY-CLASSIFICATION 'string' [FROM date]
    [, 'string' [FROM date]]...

SEE member-name [FOR 'string'] [, member-name [FOR 'string']]...
```

DATAMANAGER COMMANDS: SYNTAX

For each command, in addition to its syntax, the number of the page in the Commands Summary section on which its purposes may be found is given.

ADD

9

```
ADD member-name [NUMBER integer] {;|.}

member-definition
```

ALSO KEEP

9

```
ALSO [KEEP] [AND] command {;|.}
```

where command is one of the following DATAMANAGER commands:

```
BULK    GLOSSARY    LIST    PERFORM    WHAT    WHICH    WHOSE
```

ALTER

9

```
ALTER member-name [RENUMBER [integer]] [{NOPRINT|NO-PRINT|PRINT}] {;|.}

[amendment-line]...

{ACCEPT|QUIT|END|;|.}
```

where amendment-line is:

```
{m|LINE} [source-line]

DELETE {m [{THRU|TO} n]|LINE}

{MOVE|COPY} {m [THRU n]|LINE} {{BEFORE|TO|AFTER} {r|LINE}

CHANGE [{m [THRU n]|LINE}] [ALL] 'string' [TO] 'string-2' [PRINT]

FILL {{m [THRU n]|LINE} [COLUMNS a [THRU b]]|COLUMNS a [THRU b]} 'string' [PRINT]

BLANK {{m [THRU n]|LINE} [COLUMNS a [THRU b]]|COLUMNS a [THRU b]} [PRINT]

DISPLAY [{m [THRU {n|END}]|LINE}]

LOCATE {m [THRU {n|END}]] 'string'|NEXT ['string']|ALL 'string'}

RENUMBER m THRU n [FROM v] IN i
```

where:

m, n, r and v are source record line numbers

a and b are character positions in a source record line

i is an integer of increment.

AUTHORITY

9

```
AUTHORITY {'password'|password} {;|.}
```


BULK

9

```
BULK {ENCODE|PRINT} [status-related-selection] [selection]
     [ALPHABETICALLY] [name-related-selection]
     [time-and-user-related-selection]
     {;|.}

BULK REPORT [HIERARCHY] [ALL-STATUSES] [{SKIP|NOSKIP|NO-SKIP}]
     [status-related-selection] [selection]
     [ALPHABETICALLY] [name-related-selection]
     [time-and-user-related-selection]
     [DOWN-TO member-type]
     {;|.}
```

where

status-related-selection, name-related-selection, time-and-user-related-selection: are as specified commencing on page 33

selection is:

```
[KEPT-DATA] [MEMBERS member-name [, member-name]...]

[{[{DUMMY|EXCEPT}] member-type [, member-type]...
 |{DUMMYS|DUMMIES} [EXCEPT member-type [, member-type]...]
 |SOURCES}]
```

* SOURCES is not valid for BULK REPORT

member-type is:

```
ITEMS
GROUPS
FILES
CONVENTIONAL-FILES
DATABASES
databases
MODULES
PROGRAMS
SYSTEMS
COMMAND-STREAMS
FORMATS
OBSOLETE-DEFINITIONS
user-defined
EXECUTIVE-ROUTINES
INFOBANK-PANELS
GLOBAL-PROFILES
LOGON-PROFILES
PROFILES
```

where user-defined is a member-type in the dictionary's UDS table.

In the DOWN-TO clause, member-type must be in a lower hierarchical relation to member-name's member-type in the dictionary.

* CONVENTIONAL-FILES and OBSOLETE-DEFINITIONS do not apply in the DOWN-TO clause

COPY

9

```
COPY member-name {TO member-name-2|FROM 'status-name'} [REPLACE] {;|.}
```


DICTIONARY

9

```
DICTIONARY dictionary-name [{READ|UPDATE}] [SYSNAME file-name]
     [IBUF n] [SBUF n] [DBUF n] {;|.}
```

DOES

9

```
DOES member-name {USE member-name-2 [DIRECTLY]|DIRECTLY USE member-name-2} {;|.}
```

DROP

9

```
DROP [AND] command {;|.}
```

where command is one of the following DATAMANAGER commands:

```
BULK          WHAT
GLOSSARY      WHICH
LIST          WHOSE
PERFORM
```

ENCODE

9

```
ENCODE member-name [, member-name] ... {;|.}
```

FORMAT

9

```
FORMAT AS preferred-set-layout-name report-command {;|.}
```

where

preferred-set-layout-name is one of:

```
PSR-ITEM        PSR-PROGRAM
PSR-GROUP       PSR-SYSTEM
PSR-FILE        PSR-SUMMARY
PSR-MODULE      PSR-DETAIL
```

report-command is a DATAMANAGER REPORT or BULK REPORT command

GLOSSARY

9

```
GLOSSARY [status-related-selection] [selection]
     [ALPHABETICALLY] [name-related-selection]
     [time-and-user-related-selection] [DETAILS-ONLY]
     {;|.}

     {{GIVING|GIVE} extract [, extract]...
     |FOR clause [FOR clause]...}

[status-related-selection] [KEPT-DATA]

     {ALIASES [{alias-type|number}]] [ALPHABETICALLY]
     |alias-type ALIASES}

     [name-related-selection]
     [time-and-user-related-selection] [DETAILS-ONLY]

     [{{GIVING|GIVE} extract [, extract]...
     |FOR for-clause [FOR for-clause]...}]
```

where

status-related-selection
name-related-selection
time-and-user-related-selection
} are as specified commencing on page 33

selection is:

[KEPT-DATA] [MEMBERS member-name [, member-name]...]

[[[DUMMY | EXCEPT] category [, category]...]
  [DUMMYS | DUMMIES] [EXCEPT category [, category]...]]

where category is:

ALIASES [alias-type | number]
alias-type ALIASES
ITEMS
GROUPS
FILES
CONVENTIONAL-FILES
DATABASES
databases
MODULES
PROGRAMS
SYSTEMS
COMMAND-STREAMS
FORMATS
OBSOLETE-DEFINITIONS
user-defined
EXECUTIVE-ROUTINES
INFOBANK-PANELS
GLOBAL-PROFILES
LOGON-PROFILES
PROFILES

where user-defined is a member-type in the dictionary's UDS table.

extract is:

ALIASES [alias-type | number]
alias-type ALIASES
ACCESS-AUTHORITY
CATALOGS
CATALOGUES
EFFECTIVE-DATE
FREQUENCY
OBSOLETE-DATE
SECURITY-CLASSIFICATION
SEE
ADMINISTRATIVE-DATA | COMMENT | DESCRIPTIONS | NOTES | QUERY [LINES n [TO m | END]]
[DIRECT] REFERENCES | USAGES

FOR clause is:

FOR member-type GIVING | GIVE detail [, detail]...

where

member-type is:

ITEMS
GROUPS
FILES
CONVENTIONAL-FILES
DATABASES
databases
MODULES
PROGRAMS
SYSTEMS
COMMAND-STREAMS
FORMATS
OBSOLETE-DEFINITIONS
user-defined
EXECUTIVE-ROUTINES
INFOBANK-PANELS
GLOBAL-PROFILES
LOGON-PROFILES
PROFILES

where user-defined is a member-type in the dictionary's UDS table.

detail can be:

for any member-type: extract as defined above

for CONVENTIONAL-FILES or, if the dictionary's UDS table does not include any databases member types, for FILES:

GENERATION-CYCLE
RETENTION-PERIOD
GROWTH-RATE
DENSITY
VOLUME
SIZE

for MODULES, PROGRAMS or SYSTEMS:

AUTHOR
INSTALLATION
DATE-WRITTEN
SOURCE-COMPUTER
OBJECT-COMPUTER
SPECIAL-NAMES
I-O-CONTROL
ASSIGNMENT
EDIT-INPUT
EDIT-OUTPUT
EDIT-UPDATE

for FORMATS or any CONTROLMANAGER member-type: any clause identifier keyword appropriate to the member-type.

* user-defined keywords are available only if the User Defined Syntax facility is installed.
OBSOLETE-DEFINITIONS is available only if the Status facility is installed.
EXECUTIVE-ROUTINES is available only if the User Defined Commands facility is installed.
INFOBANK-PANELS is available only if the User Defined InfoSystem facility is installed.

INSERT

9

INSERT member-name [NUMBER integer] [NOPRINT | NO-PRINT | PRINT] { ; | . }

member-definition

B26

KEEP

10

{ KEEP | ALSO [KEEP] } [AND] command { ; | . }

where command is one of the following DATAMANAGER commands:

BULK
GLOSSARY
LIST
PERFORM
WHAT
WHICH
WHOSE

LIST

10

LIST [HISTORY] [status-related-selection] [selection]
[ALPHABETICALLY] [name-related-selection]
[time-and-user-related-selection] [DETAILS-ONLY] { ; | . }

where

status-related-selection
name-related-selection
time-and-user-related-selection
} are as specified commencing on page 33

selection is:

[KEPT-DATA] [MEMBERS member-name [, member-name]...]

[[[DUMMY | EXCEPT] index-name-type [, index-name-type]...]
  [DUMMYS | DUMMIES | INDEX-NAMES] [EXCEPT index-name-type [, index-name-type]...]
  SOURCES]

where index-name-type is:

ITEMS
GROUPS
FILES
CONVENTIONAL-FILES
DATABASES
databases
MODULES
PROGRAMS
SYSTEMS
COMMAND-STREAMS
FORMATS
OBSOLETE-DEFINITIONS
ALIASES
CATALOGS
CATALOGUES
ATTRIBUTES
user-defined
EXECUTIVE-ROUTINES
INFOBANK-PANELS
GLOBAL-PROFILES
LOGON-PROFILES
PROFILES

where user-defined is a member-type in the dictionary's UDS table.

* user-defined keywords are available only if the User Defined Syntax facility is installed.
OBSOLETE-DEFINITIONS is available only if the Status facility is installed.
EXECUTIVE-ROUTINES is available only if the User Defined Commands facility is installed.
INFOBANK-PANELS is available only if the User Defined InfoSystem facility is installed.

B2c

MODIFY

10

```
MODIFY member-name [RENUMBER [integer]] {;}
                                        {.}

[amendment-line]...

{ACCEPT}
{QUIT  }
{END   }
{;     }
{.     }
```

where amendment-line is:

```
{m    } [source-line]
{LINE }

DELETE {m [{THRU} n]}
       {   {TO  }   }
       {LINE        }

{MOVE}  {m [THRU n]} {BEFORE} {r    }
{COPY}  {LINE      } {TO    } {LINE }
                     {AFTER }

CHANGE [{m [THRU n]}] [ALL] 'string' [TO] 'string-2' [PRINT]
       {LINE       }

FILL {{m [THRU n]} [COLUMNS a [THRU b]]} 'string' [PRINT]
     {LINE                             }
     {COLUMNS a [THRU b]               }

BLANK {{m [THRU n]} [COLUMNS a [THRU b]]} [PRINT]
      {LINE                             }
      {COLUMNS a [THRU b]               }

DISPLAY [{m [THRU {n  }]}]
         {        {END}  }
         {LINE           }

LOCATE {[m [THRU {n  }]] 'string'}
       {         {END}           }
       {NEXT ['string']          }
       {ALL 'string'             }

RENUMBER m THRU n [FROM v] IN i
```

where:

r, n, r and v are source record line numbers

a and b are character positions in a source record line

i is an integer of increment.

PERFORM

10

```
PERFORM {'input-line' [, 'input-line']...}
        {COMMAND-STREAM member-name      }

    [status-related-selection] [selection]
    [ALPHABETICALLY] [name-related-selection]
    [time-and-user-related-selection]

    [{NO-TERMINATOR}] [CLEAR-KEPT-DATA] [{PRINT [FIRST]}] {;}
     {NOTERMINATOR }                    {DISPLAY-ONLY  } {.}
```

where

input-line is any valid input line and may contain an asterisk where a member-name, alias, catalog classification or user defined indexed attribute would normally occur

status-related-selection
name-related-selection
time-and-user-related-selection

are as specified commencing on page 33

selection is:

```
[KEPT-DATA] [MEMBERS member-name [, member-name]...]

[{{DUMMY } index-name-type [, index-name-type]...}]
 {{EXCEPT}                                       }

 {{DUMMYS    }                                   }
 {{DUMMIES   } [EXCEPT index-name-type           }
 {{INDEX-NAMES}      [, index-name-type]...      }

 {SOURCES                                        }
```

where index-name-type is:

```
{ITEMS                 }
{GROUPS                }
{FILES                 }
{CONVENTIONAL-FILES    }
{DATABASES             }
{databases             }
{MODULES               }
{PROGRAMS              }
{SYSTEMS               }
{COMMAND-STREAMS       }
{FORMATS               }
{OBSOLETE-DEFINITIONS  }
{ALIASES               }
{CATALOGS              }
{CATALOGUES            }
{ATTRIBUTES            }
{user-defined          }
{EXECUTIVE-ROUTINES    }
{INFOBANK-PANELS       }
{GLOBAL-PROFILES       }
{LOGON-PROFILES        }
{PROFILES              }
```

where user-defined is a member-type in the dictionary's UDS table.

- user-defined keywords are available only if the User Defined Syntax facility is installed.
  OBSOLETE-DEFINITIONS is available only if the Status facility is installed.
  EXECUTIVE-ROUTINES is available only if the User Defined Commands facility is installed.
  INFOBANK-PANELS is available only if the User Defined InfoSystem facility is installed.

PRINT

10

```
PRINT {member-name [, member-name]...                          } {;}
      {member-name LINES {integer                              } {.}
                         {line-number TO {line-number-2}       }
                         {               {END           }      }
```

REMOVE

10

```
REMOVE member-name [, member-name]... {;}
                                      {.}
```

RENAME

10

```
RENAME member-name AS member-name-2 {;}
                                    {.}
```

REPLACE

10

```
REPLACE member-name [NUMBER interger] {;}
                                      {.}

member-definition
```


REPORT

10

```
REPORT member-name [ALL-STATUSES]

    [{HIERARCHY          }] [{SKIP   }]
     {DOWN-TO member-type}  {NO-SKIP}
                            {NOSKIP }

    [, member-name [ALL-STATUSES]

        [{HIERARCHY          }] [{SKIP   }]]... {;}
         {DOWN-TO member-type}  {NO-SKIP}       {.}
                                {NOSKIP }
```

where member-type is:

```
{ITEMS                 }
{GROUPS                }
{FILES                 }
{DATABASES             }
{databases             }
{MODULES               }
{PROGRAMS              }
{SYSTEMS               }
{COMMAND-STREAMS       }
{FORMATS               }
{user-defined          }
{EXECUTIVE-ROUTINES    }
{INFOBANK-PANELS       }
{GLOBAL-PROFILES       }
{LOGON-PROFILES        }
{PROFILES              }
```

where user-defined is a member-type in a (lower) hierarchical relation to member-name's member-type in the dictionary's UDS table

SHOW ALIAS-TYPES

11

```
SHOW ALIAS-TYPES {;}
                 {.}
```

SHOW UDS

11

```
SHOW UDS [FOR MSP]

         {FULL                                  } {;}
         {MEMBER-TYPES [clause-1]               } {.}
         {USER-DEFINED ATTRIBUTES [clause-2]    }
```

where

clause-1 is:

```
[[{ENCODE       } [, {ENCODE       }]...] KEYWORDS]
  {INTERROGATE  }    {INTERROGATE  }
  {REPORT-DOWN-TO}   {REPORT-DOWN-TO}

[[{STANDARD} [, {STANDARD}]...] LITERALS]
  {SHORT   }    {SHORT   }
  {LONG    }    {LONG    }
  {PLURAL  }    {PLURAL  }

[HIERARCHY]

[RELATIONSHIPS [FOR member-type [,member-type]...]]
```

clause-2 is:

```
 [FOR member-type [,member-type]...]
 [IDENTIFIED-BY attribute-id [,attribute-id]...]
 [GIVING DETAILS]
```

The USER-DEFINED ATTRIBUTES clause is available only if the User Defined Syntax facility is installed.

WHAT 11

```
WHAT  { USES         } member-name [DIRECTLY]                          } {;}
      { CONSTITUTES  }                                                 } {.}
      |
      | DIRECTLY { USES         } member-name
      |        { CONSTITUTES  }
      |
      | FORMS 'classification' [, 'classification']...
      |
      |     [{AND} [(]... [[DOES] NOT] [(]... condition
      |      {OR }
      |           [{AND} [(]... [[DOES] NOT] [(]...
      |            {OR }          condition [)]...]...]
      |
      | IS {name   }
      |    {'name' }
      |
      | BELONGS-TO 'owner-name'
```

where condition is as specified for the WHICH command.

WHICH 11

```
WHICH [status-related-selection] selection
    [ALPHABETICALLY] [name-related-selection]
    [time-and-user-related-selection]

    [(]... [[DOES] NOT] [(]... condition
    [{AND} [(]... [[DOES] NOT] [(]... condition [)]...]... {;}
     {OR }                                                 {.}
```

where

status-related-selection
name-related-selection
time-and-user-related-selection
} are as specified commencing on page 33

selection is:

```
{MEMBERS                                                    }
{KEPT-DATA [[EXCEPT] category [, category]...]              }
{[EXCEPT] category [, category]...                          }
{INDEX-NAMES [EXCEPT category [, category]...]              }
```

where category is:

```
{ITEMS                                  }
{GROUPS                                 }
{FILES                                  }
{CONVENTIONAL-FILES                     }
{DATABASES                              }
{databases                              }
{MODULES                                }
{PROGRAMS                               }
{SYSTEMS                                }
{COMMAND-STREAMS                        }
{FORMATS                                }
{OBSOLETE-DEFINITIONS                   }
{user-defined                           }
{EXECUTIVE-ROUTINES                     }
{INFOBANK-PANELS                        }
{GLOBAL-PROFILES                        }
{LOGION-PROFILES                        }
{PROFILES                               }
{ALIASES [{alias-type}]                 }
{        {number    }                   }
{alias-type ALIASES                     }
{CATALOGS                               }
{CATALOGUES                             }
{ATTRIBUTES                             }
```

where user-defined is a member-type in the dictionary's UDS table.

condition is:

```
[DIRECTLY]

{ {USES        } member-name [form] [VIA {INPUTS      }]
{ {CONSTITUTES }                         {OUTPUTS     }
{                                        {UPDATES     }
{                                        {CONTAINS    }
{                                        {CALLS       }
{                                        {PASSING     }
{                                        {PARAMETERS  }
{                                        {KEYS        }
{                                        {SORT-KEYS   }
{                                        {SEE         }
{                                        {USER-EXIT   }
{                                        {USER-LABELS }
{                                        {IF          }
{                                        {NAME        }
{                                        {BOUND       }
{
{ {INPUTS      } member-name [form]
{ {OUTPUTS     }
{ {UPDATES     }
{ {CONTAINS    }
{ {CALLS       }
{ {PASSES      }
{ {INPUT-BY    }
{ {OUTPUT-BY   }
{ {UPDATED-BY  }
{ {CONTAINED-BY}
{ {CALLED-BY   }
{ {PASSED-BY   }

    [DIRECTLY]

FORMS 'classification' [, 'classification']...

{HAS } attribute-specification
{HAVE}

{BELONGS-TO} 'owner-name'
{BELONG-TO }
```

where

form is:

```
{ [{ENTERED-AS}] [[VERSION] version]  }
{  {HELD-AS   }                       }
{  {REPORTED-AS}                      }
{                                     }
{ DEFAULTED-AS                        }
```

attribute-specification is:

```
{attribute-id [FOR member-type] {relation  }                }
{                               {SPECIFIED }                }
{                                                           }
{    [FOR member-type {relation  }]...                      }
{                     {SPECIFIED }                          }
{                                                           }
{clause-id [FOR member-type] SPECIFIED                      }
{   [FOR member-type SPECIFIED]...                          }
```


where

attribute-id is:

```
{ADMINISTRATIVE-DATA                    }
{ALIAS [{alias-type}]                   }
{      {number    }                     }
{alias-type ALIAS                       }
{CATALOGUE                              }
{COMMENT                                }
{DESCRIPTION                            }
{EFFECTIVE-DATE                         }
{NOTE                                   }
{OBSOLETE-DATE                          }
{QUERY                                  }
{GENERATION-CYCLE                       }
{DENSITY                                }
{VOLUME                                 }
{AUTHOR                                 }
{INSTALLATION                           }
{DATE-WRITTEN                           }
{SOURCE-COMPUTER                        }
{OBJECT-COMPUTER                        }
{SPECIAL-NAMES                          }
{I-O-CONTROL                            }
{ASSIGNMENT                             }
{EDIT-INPUT                             }
{EDIT-OUTPUT                            }
{EDIT-UPDATE                            }
```

clause-id is:

```
{ACCESS-AUTHORITY          }
{FREQUENCY                 }
{SECURITY-CLASSIFICATION   }
{SEE                       }
{RETENTION-PERIOD          }
{GROWTH-RATE               }
{SIZE                      }
```

member-type is:

```
{ITEMS                          }
{GROUPS                         }
{FILES                          }
{CONVENTIONAL-FILES             }
{DATABASES                      }
{databases                      }
{MODULES                        }
{PROGRAMS                       }
{SYSTEMS                        }
{COMMAND-STREAMS                }
{FORMATS                        }
{OBSOLETE-DEFINITIONS           }
{user-defined                   }
{EXECUTIVE-ROUTINES             }
{INFOBANK-PANELS                }
{GLOBAL-PROFILES                }
{LOGON-PROFILES                 }
{PROFILES                       }
```

where user-defined is a member-type in the dictionary's UDS table.

relation is:

```
{operator value [, value]...                                }
{                                                           }
{[LINES n [TO {m  }]]                                       }
{             {END}                                         }
{ {INCLUDING} 'string' [, 'string']...                      }
{ {INCLUDES }                                               }
```

where operator is:

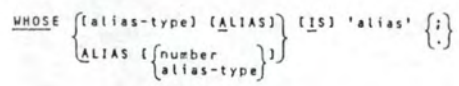
```
{EQUALS }
{=      }
{NE     }
{GT     }
{>      }
{GE     }
{LT     }
{<      }
{LE     }
```

The VIA clause of condition can accept a keyword other than one of those listed above, if the keyword identifies a clause, appropriate to a member type of any installed MANAGER Product nucleus or selectable unit, that sets up a relationship.

(USES/CONSTITUTES conditions appear as "relation-clause" on DBMS interface syntax cards.)

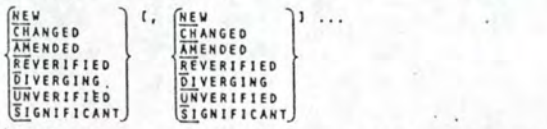
* user-defined keywords are available only if the User Defined Syntax is installed.
OBSOLETE-DEFINITIONS is available only if the Status facility is installed.
EXECUTIVE-ROUTINES is available only if the User Defined Commands facility is installed.
INFOBANK-PANELS is available only if the User Defined InfoSystem facility is installed.

WHOSE 11



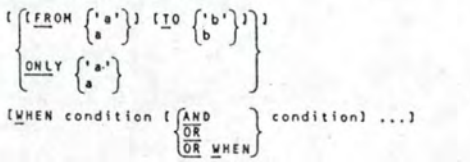
Secondary Selection Keywords and Clauses

status-related-selection is:

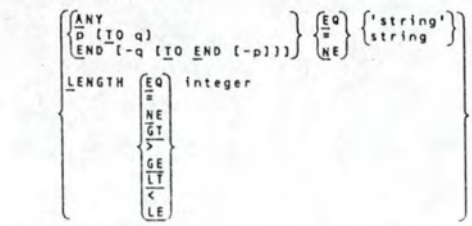


SIGNIFICANT is applicable only to the GLOSSARY command

name-related-selection is:

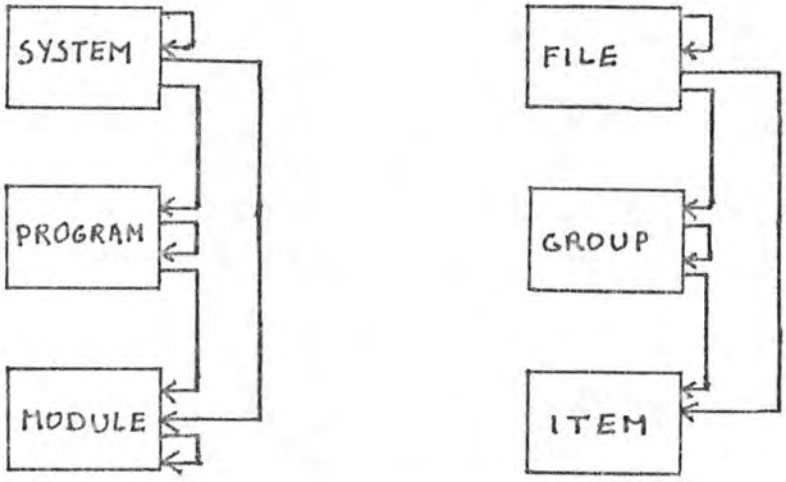


where condition is:

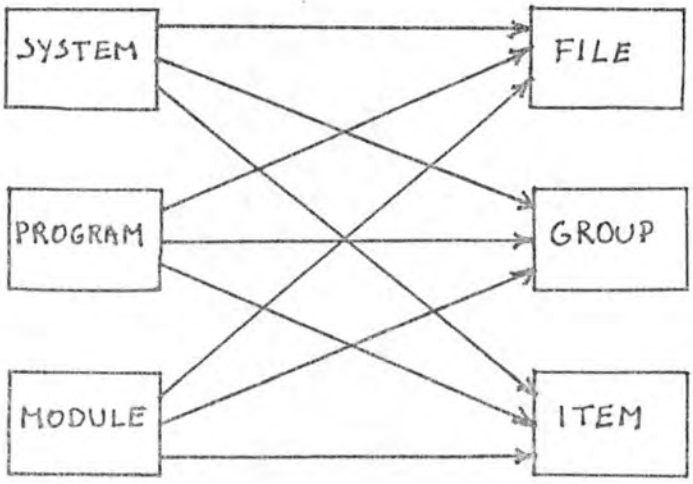


REPRESENTATION DU DOMAINE D'APPLICATION DES RELATIONS

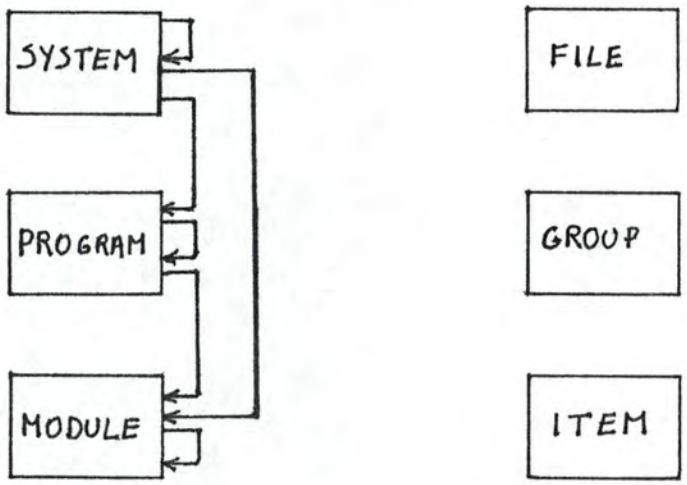
Relation CONTAINS (membre)



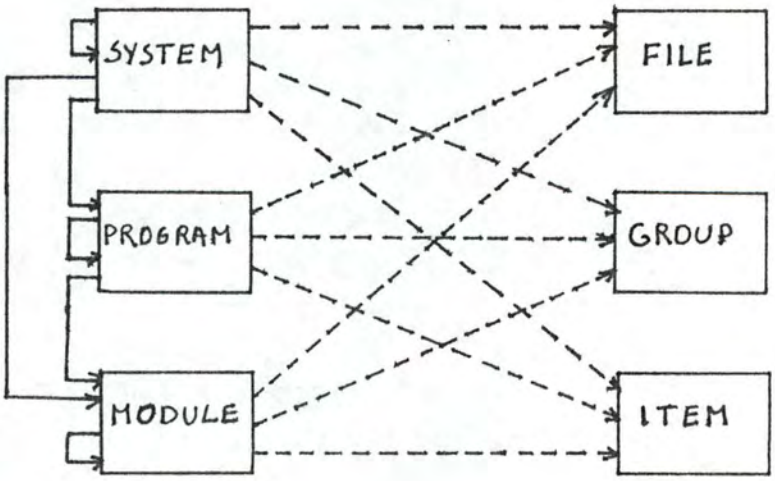
Relations INPUTS, OUTPUTS, UPDATES (membre donnée)



Relation CALLS (membre traitement)

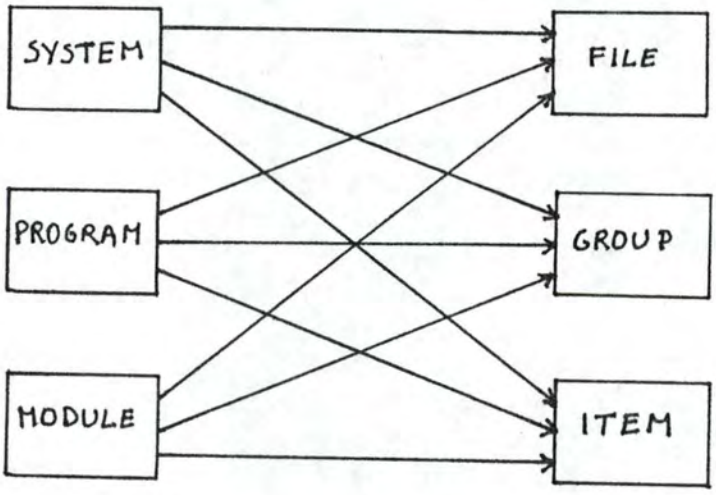


Relation CALLS (membre traitement) PASSING (membre donnée) AT 'chaîne de caractères'.



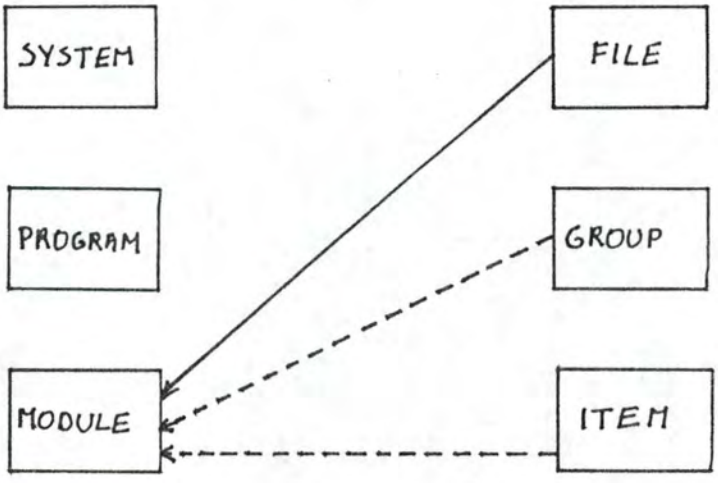
Relation PARAMETERS (membre donnée)

ENTRY-POINT 'chaine de caractères'

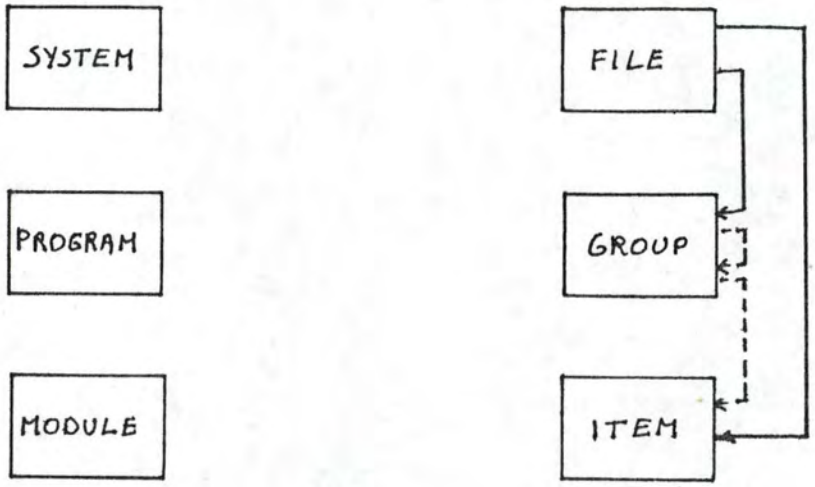


Relations USER-LABEL (membre module) et

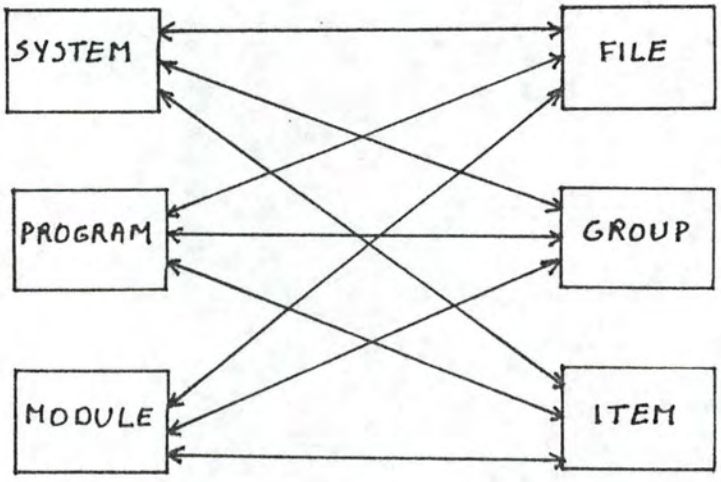
USER_EXIT (membre module)



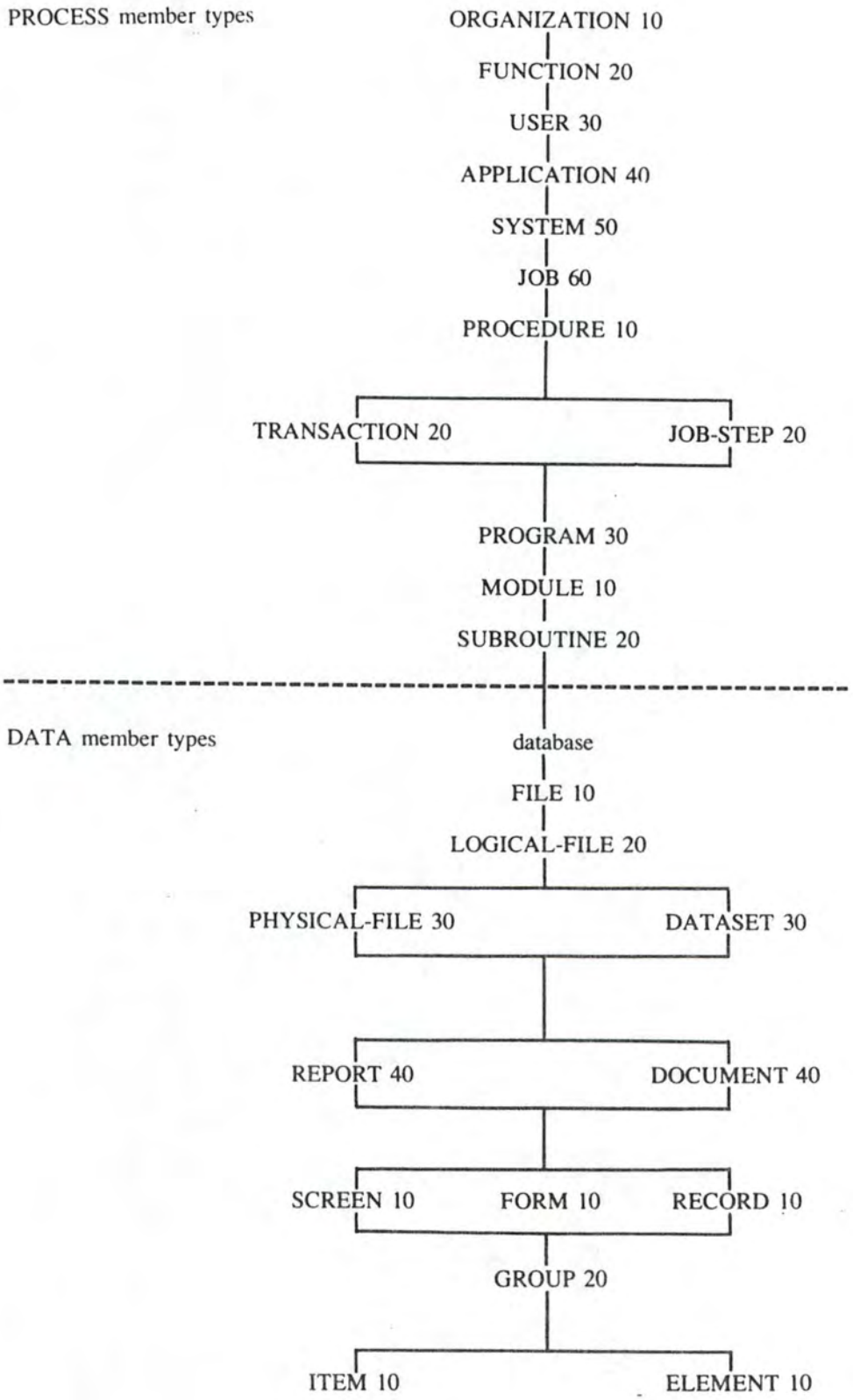
Relations SORT-KEY (membre traitement) et KEYS (membre traitement)



Relation SEE (membre) FOR 'chaîne de caractères'

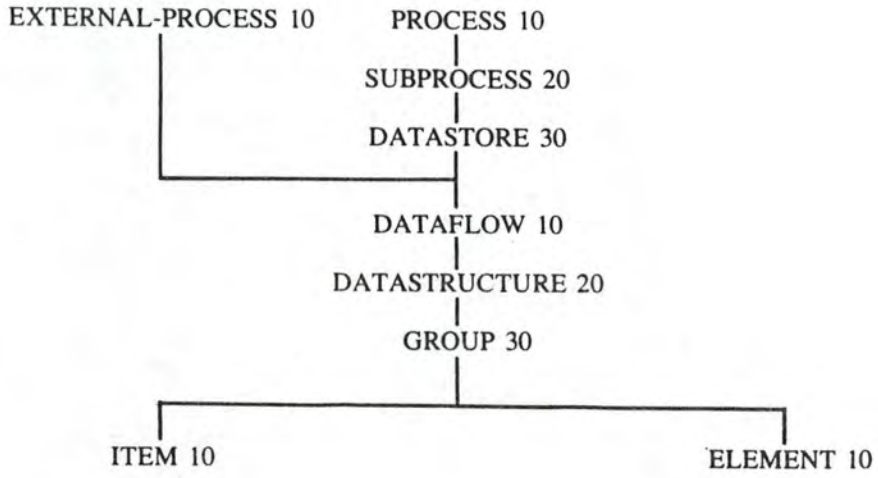


The EDPS Hierarchy



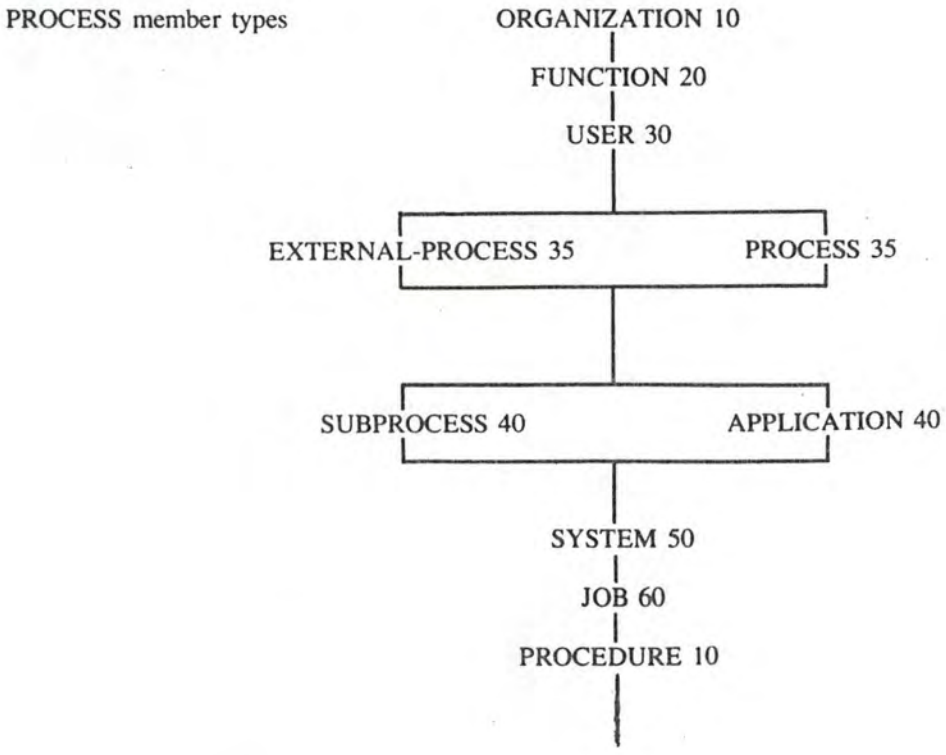
The SAS Hierarchy

SAS is a member type set which represents terms in "top down" or "structured design" techniques as promoted by Gane and Sarson of IST Inc., or by Yourdon Inc.

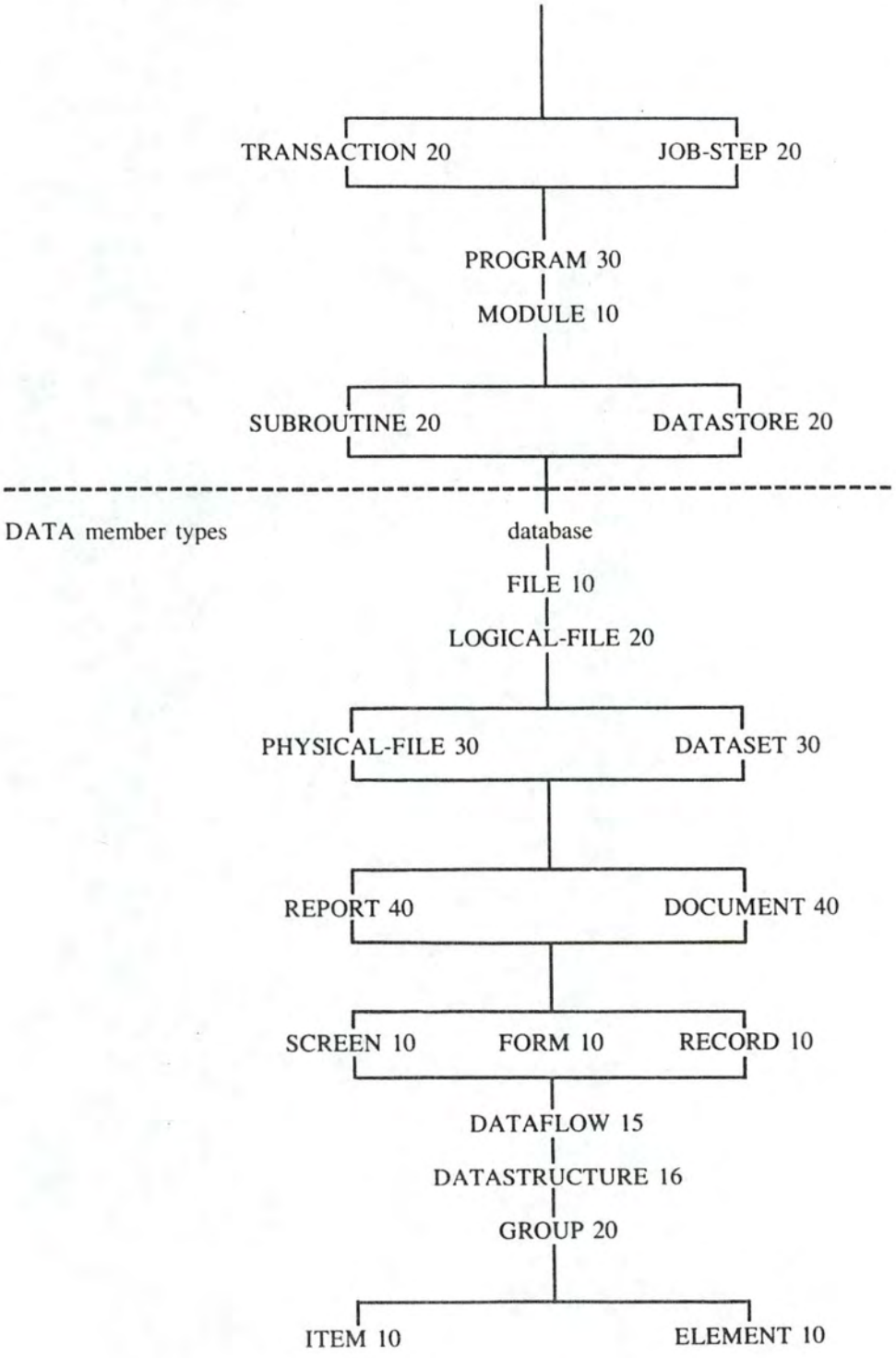


The SDS Hierarchy

The SDS hierarchy is a combination of the EDPS and SAS member type hierarchies.

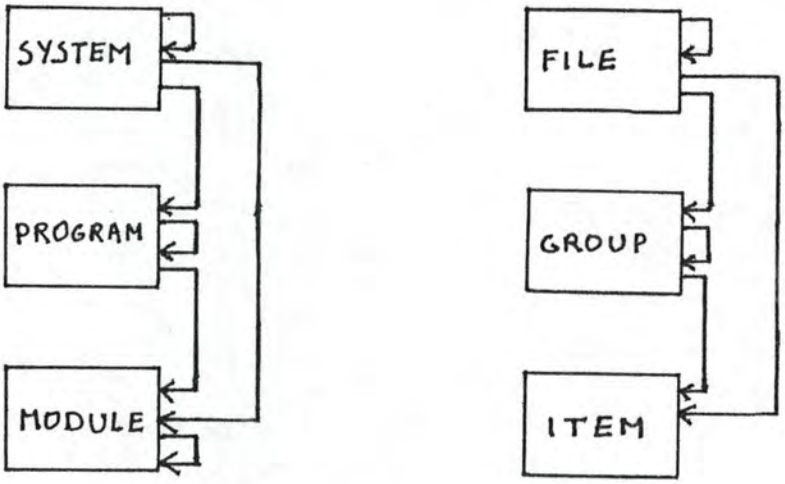


B4c

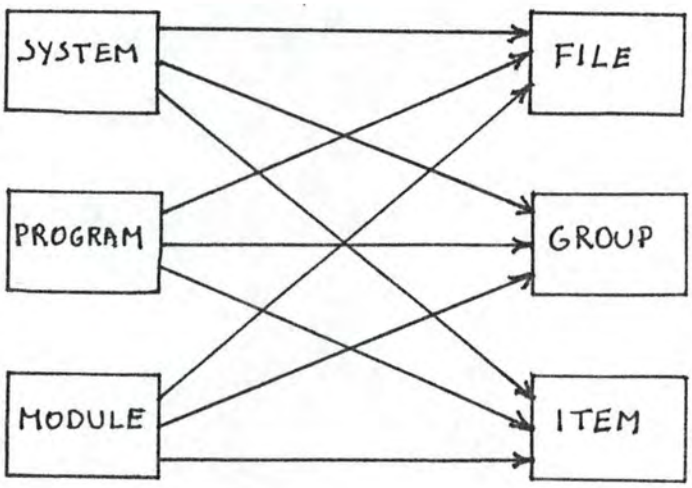

TRANSACTION 20
JOB-STEP 20
PROGRAM 30
MODULE 10
SUBROUTINE 20
DATASTORE 20
DATA member types
database
FILE 10
LOGICAL-FILE 20
PHYSICAL-FILE 30
DATASET 30
REPORT 40
DOCUMENT 40
SCREEN 10
FORM 10
RECORD 10
DATAFLOW 15
DATASTRUCTURE 16
GROUP 20
ITEM 10
ELEMENT 10

REPRESENTATION DU DOMAINE D'APPLICATION DES RELATIONS

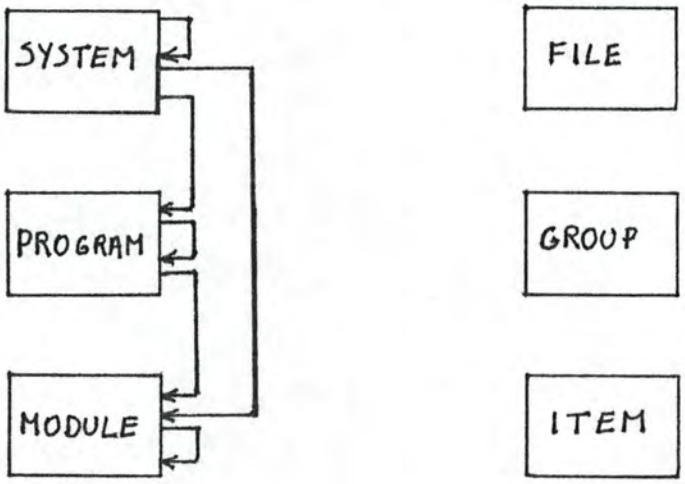
Relation CONTAINS (membre)



Relations INPUTS, OUTPUTS, UPDATES (membre donnée)

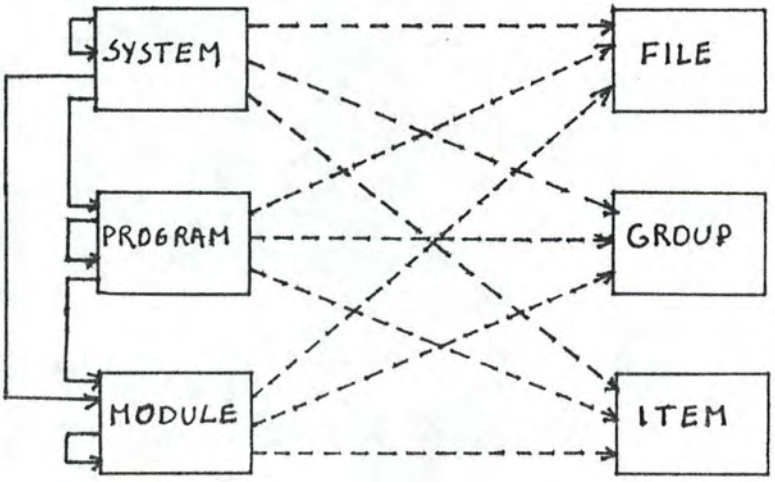


Relation CALLS (membre traitement)



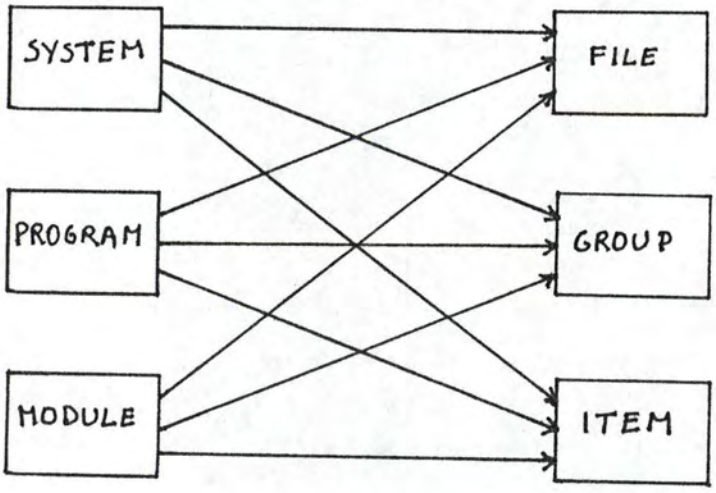
Relation CALLS (membre traitement) PASSING (membre donnée)

AT 'chaîne de caractères'.



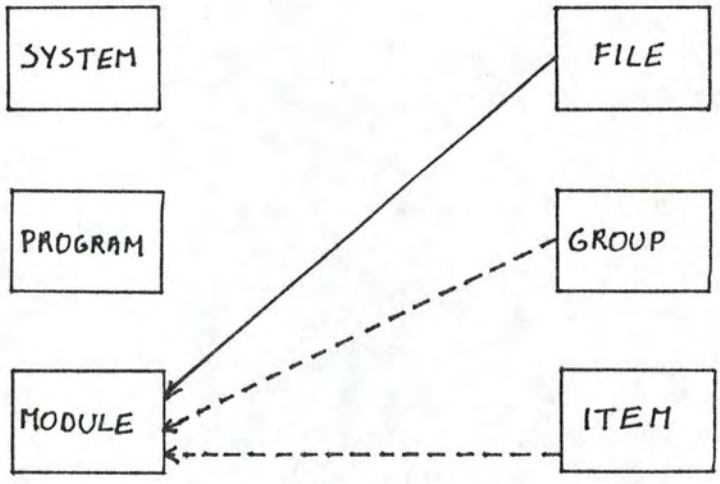
Relation PARAMETERS (membre donnée)

ENTRY-POINT 'chaine de caractères'

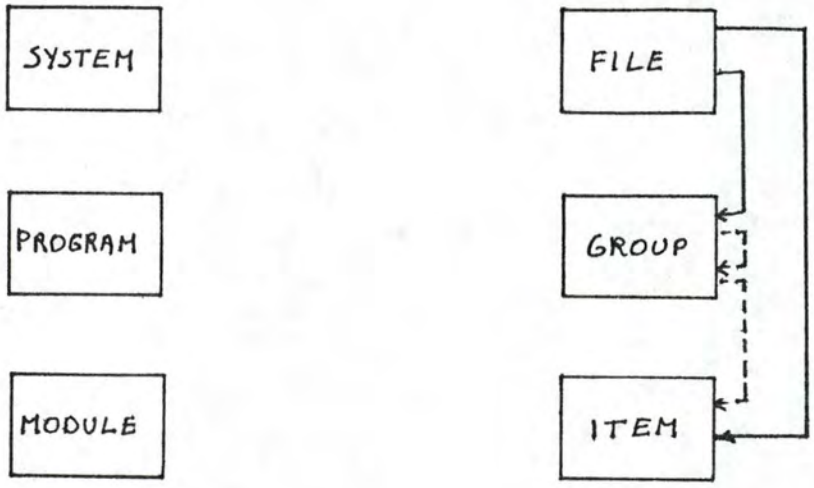


Relations USER-LABEL (membre module) et

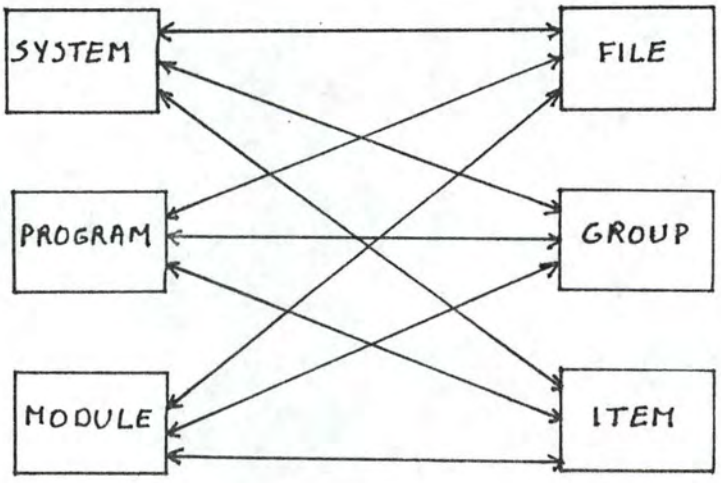
USER_EXIT (membre module)



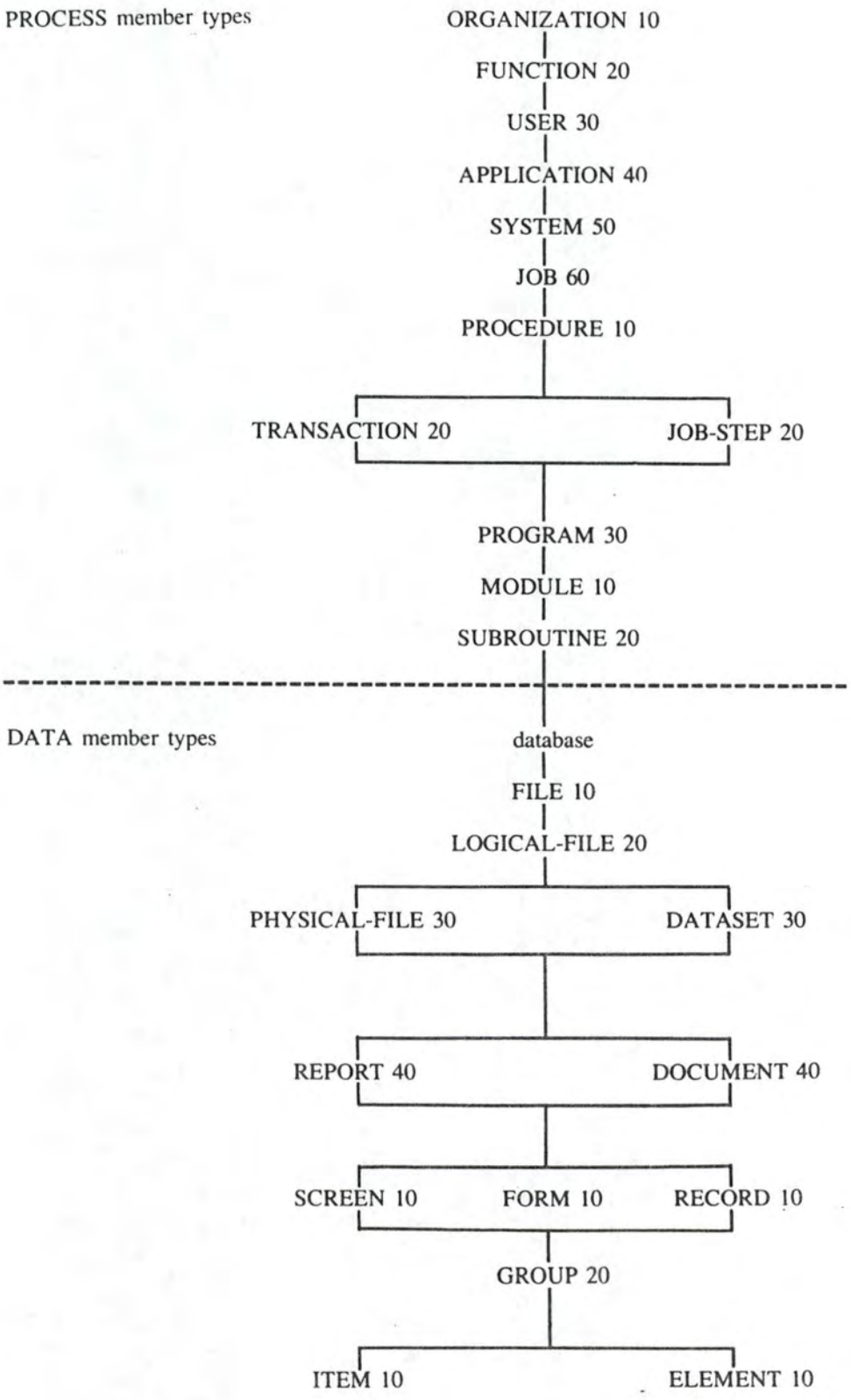
Relations SORT-KEY (membre traitement) et KEYS (membre traitement)



Relation SEE (membre) FOR 'chaîne de caractères'



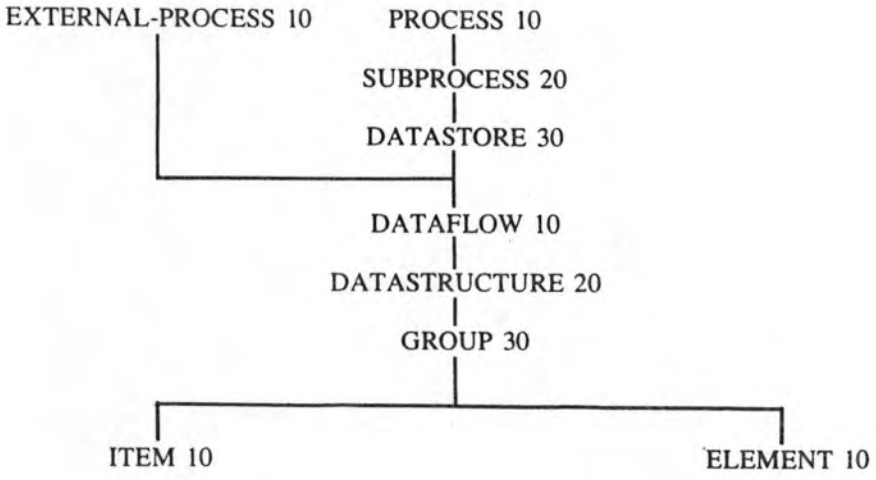
The EDPS Hierarchy



MSP
DATAMANAGER

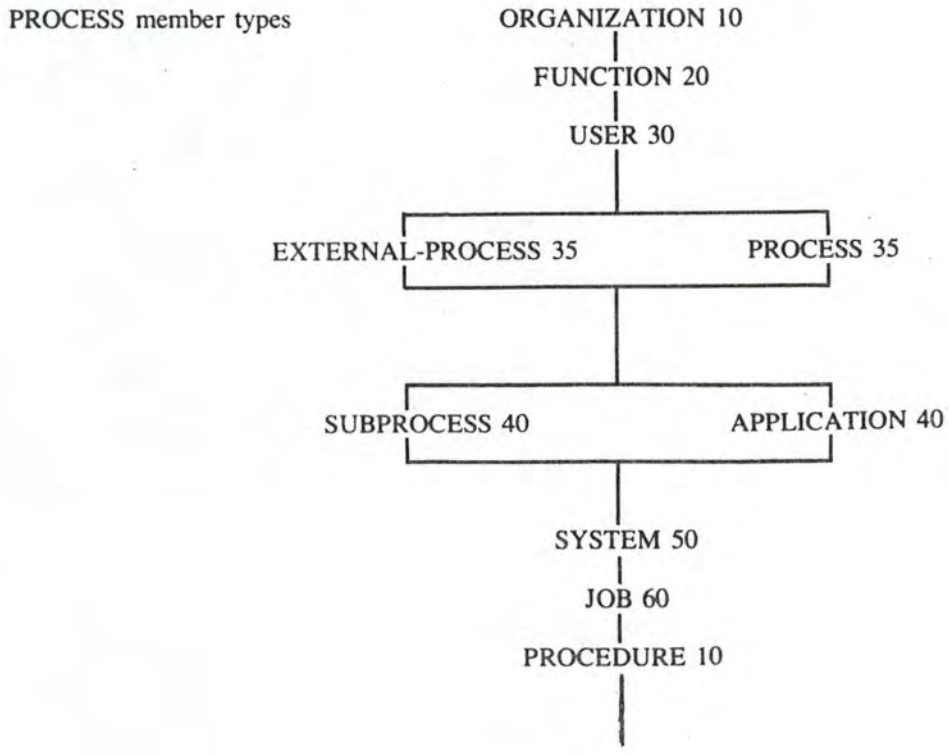
The SAS Hierarchy

SAS is a member type set which represents terms in "top down" or "structured design" techniques as promoted by Gane and Sarson of IST Inc., or by Yourdon Inc.

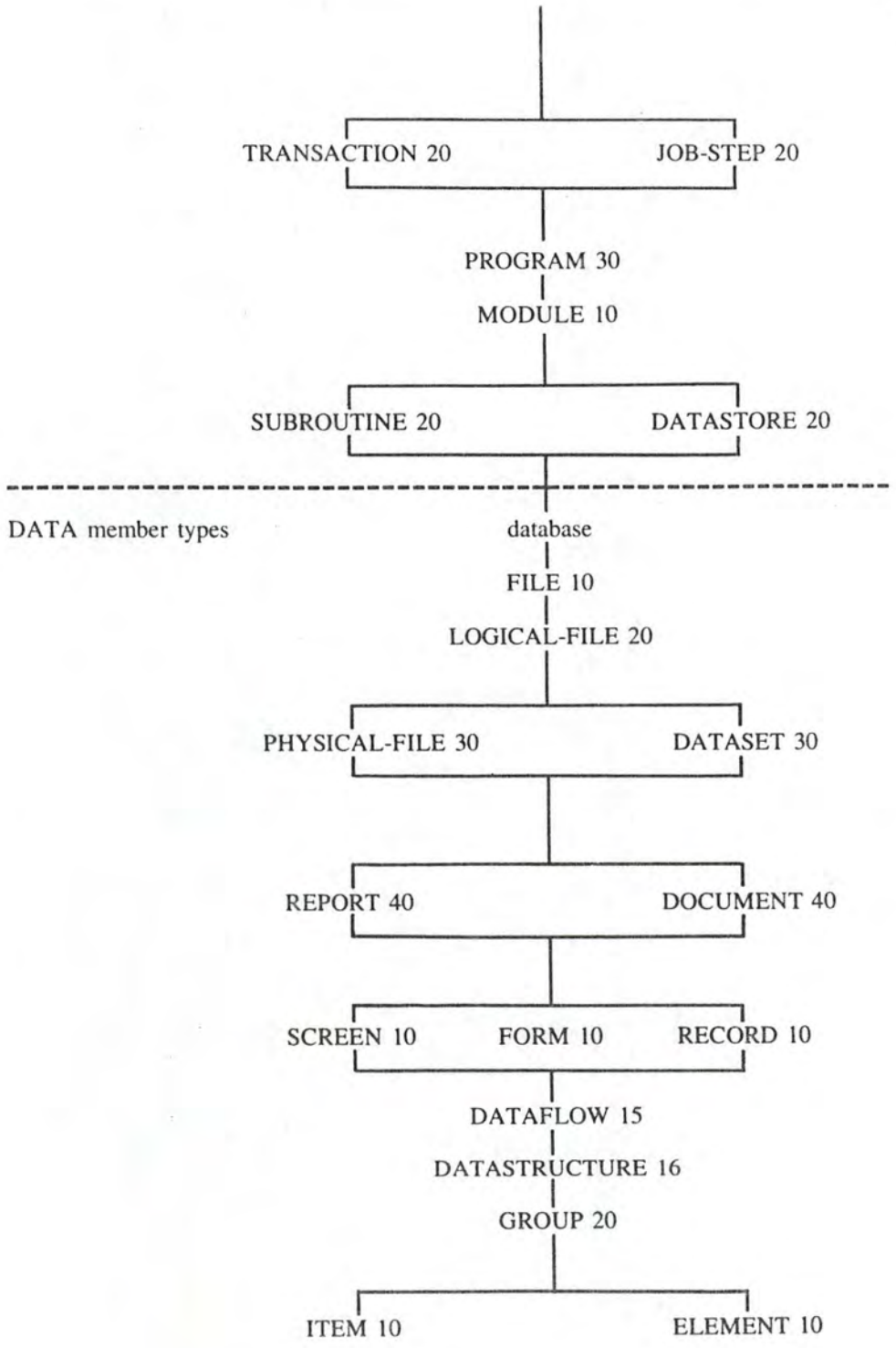


The SDS Hierarchy

The SDS hierarchy is a combination of the EDPS and SAS member type hierarchies.



B4c


TRANSACTION 20
JOB-STEP 20
PROGRAM 30
MODULE 10
SUBROUTINE 20
DATASTORE 20
DATA member types
database
FILE 10
LOGICAL-FILE 20
PHYSICAL-FILE 30
DATASET 30
REPORT 40
DOCUMENT 40
SCREEN 10
FORM 10
RECORD 10
DATAFLOW 15
DATASTRUCTURE 16
GROUP 20
ITEM 10
ELEMENT 10