Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Contribution à l'introduction d'une méthodologie de développement dans une entreprise

Lefebvre, Christian; Philippot, Jean-Noël

Award date:
1989

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 03. Jul. 2025

CONTRIBUTION A L'INTRODUCTION D'UNE METHODOLOGIE DE DEVELOPPEMENT DANS UNE ENTREPRISE

Christian LEFEBVRE
Jean-Noël PHILIPPOT

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

Promoteur : M. Baudouin LE CHARLIER.

Année académique : 1988-1989.

Résumé

Ce mémoire contribue à l'introduction de la 'System Development Methodology' (S.D.M.) à la Caisse Générale d'Epargne et de Retraite. Nous présentons d'abord l'environnement de travail de la C.G.E.R. et, plus particulièrement, les éléments méthodologiques existants. Puis, nous présentons S.D.M. ainsi qu'une étude de différents modèles et outils susceptibles de la compléter. A partir de ces éléments, nous proposons une intégration que nous espérons utilisable par la C.G.E.R. Pour vérifier la plausibilité de notre proposition, nous l'appliquons à un cas pratique. Enfin, nous donnons quelques considérations au sujet de S.D.M. et sur les méthodes en général.

Abstract

This study contributes to the 'System Development Methodology' (S.D.M.) introduction in the C.G.E.R. First we present the existing working environment at the C.G.E.R. and especially parts of methods already in use. Then we present S.D.M. and a study on several models and tools that could make it up. From all these elements we propose an integrated method that could be used by the C.G.E.R. We also apply it to verify its plausibility. At last we give some considerations about S.D.M. and methods.

AVANT - PROPOS

La réalisation de ce mémoire est le fruit d'un travail commun de recherche et de réflexion de plusieurs mois. Néanmoins, notre personnalité nous a guidé vers un développement individuel de certaines parties de cette étude.

La répartition des tâches est la suivante :

- Le second chapitre de la première partie (Tentative de mise au point d'une méthodologie de développement utilisable dans un environnement contraigant) a été écrit par Christian.

- Les premier, troisième et quatrième chapitres de cette même partie ont été réalisés par Jean-Noël.

- La deuxième partie de ce mémoire (Application de la méthode proposée à un cas pratique) a été réalisée conjointement.

- Le premier chapitre de la troisième partie (Prise de conscience des avantages et des limites des méthodes) a été élaboré par Christian.

- Les conclusions ont été tirées par Jean-Noël.

REMERCIEMENTS

Nos premiers remerciements vont à Baudouin LE CHARLIER, promoteur de ce mémoire, pour les précieux conseils qu'il nous a prodigués, ainsi que pour son soutien pédagogique et moral, soutien ayant contribué largement à la réalisation de ce mémoire.

Nous exprimons également notre reconnaissance à Marie-Claire VERLAINE, chef du service Méthodologie et Administration de l'Information de la Caisse Générale d'Epargne et de Retraite, pour son accueil ainsi que pour le temps qu'elle nous a consacré et l'expérience dont elle nous a fait profiter.

Nous tenons aussi à exprimer notre gratitude envers Christian GRAAS et Alain SPILTOIR qui nous ont accordé de nombreuses entrevues dans le cadre de la partie théorique de ce mémoire, ainsi qu'envers Marc PETIT, Norbert BASTIEN et Guy DE LEEUW pour les conseils apportés lors du développement de l'application.

Enfin, nous remercions tous les membres du M.A.I. ainsi que toutes les personnes de la C.G.E.R. qui, de près ou de loin, ont contribué à la réalisation de ce projet.

Table des Matières

INTRODUCTION 1

PARTIE I : TENTATIVE DE MISE AU POINT D'UNE METHODOLOGIE DE DEVELOPPEMENT UTILISABLE DANS UN ENVIRONNEMENT CONTRAIGNANT (LA C.G.E.R.) 4

Introduction 5

Chapitre 1 : Présentation de l'environnement de développement de la C.G.E.R. 6

1.1. Introduction 7

1.2. Distinction préalable : cycle de vie - méthode . . 7

1.3. L'environnement de développement de la C.G.E.R. . 8

1.4. Aperçu du développement des projets informatiques 9

1.4.1. La demande de développement 10

1.4.2. La définition de projet 10

1.4.3. L'analyse conceptuelle 11

1.4.3.1. Le volet "données" 12

1.4.3.2. Le volet "traitements" 13

1.4.4. PALOMA 14

1.4.5. L'analyse fonctionnelle 15

1.4.5.1. Le volet "données" 15

1.4.5.2. Le volet "traitements" 16

1.4.6. L'analyse technique 17

1.4.7. La programmation et les tests 17

1.4.8. Le passage en production 18

1.5. Conclusion 18

Chapitre 2 : Présentation de la méthode S.D.M. 20

2.1. Remarque préliminaire 21

2.2. La méthode S.D.M. 21

2.3. Phase 1 : Etude de l'existant et Analyse conceptuelle 26
2.3.1. Description 26
2.3.2. Planning possible des activités 28
2.3.3. Description des activités 28
2.4. Phase 2 : Analyse fonctionnelle 33
2.4.1. Description 33
2.4.2. Planning possible des activités 35
2.4.3. Description des activités 36
2.5. Phase 3 : Analyse technique 40
2.5.1. Description 40
2.5.2. Planning possible des activités 42
2.5.3. Description des activités 42
2.6. Phase 4 : Réalisation des programmes 45
2.6.1. Description 45
2.6.2. Planning possible des activités 46
2.6.3. Description des activités 46
2.7. Phase 5 : Tests 49
2.7.1. Description 49
2.7.2. Planning possible des activités 51
2.7.3. Description des activités 51
2.8. Phase 6 : Conversion et mise en production . . . 54
2.8.1. Description 54
2.8.2. Planning possible des activités 55
2.8.3. Description des activités 55
2.9. Phase 7 : Utilisation et gestion du système . . 58
2.9.1. Description 58
2.9.2. Planning possible des activités 59
2.9.3. Description des activités 60
2.10. Synthèse 62
2.10.1. Pour les données 62
2.10.2. Pour les traitements 63
2.10.3. Pour la sécurité 64

Chapitre 3 : Les modèles et outils d'aide au développement de logiciels 66
3.1. Introduction 67
3.2. Distinction modèles - outils 70

3.2.1. *Les modèles* 70
3.2.2. *Les outils* 72
3.3. *Les modèles d'aide au développement de logiciels* 74
3.3.1. *La modélisation conceptuelle des données* . 74
3.3.1.1. *Définitions* 76
3.3.1.2. *Décomposition de la démarche* 77
a: Elaboration de la liste brute des informations 77
b: Epuration de la liste brute 77
c: Dégagement des types d'entités 77
d: Rattachement de leurs propriétés aux entités 78
e: Définition des relations inter-entités . 78
f: Déterminer les cardinalités 78
g: Simplifier éventuellement le modèle . . . 79
h: Quantification du modèle conceptuel . . . 79
3.3.1.3. *Domaines d'application du modèle* . . . 80
3.3.1.4. *Avantages d'un modèle E-A* 80
3.3.1.5. *Inconvénients d'un modèle E-A* 81
3.3.2. *La normalisation* 81
3.3.2.1. *Concepts de base* 82
a: La représentation des informations . . . 84
b: La première forme normale (1FN) 84
c: La deuxième forme normale (2FN) 85
d: La troisième forme normale (3FN) 85
e: La quatrième forme normale (4FN) 86
f: La cinquième forme normale (5FN) 86
3.3.2.2. *Domaine d'application de la normalisation* 86
3.3.2.3. *Avantages de la normalisation* 87
3.3.2.4. *Inconvénients de la normalisation* . . 87
3.3.3. *Le modèle dynamique des données* 88
3.3.4. *Le modèle logique des données* 88
3.3.4.1. *Les objets de base du modèle* 89
3.3.4.2. *Domaine d'application du M.A.G.* . . . 91
3.3.4.3. *Evaluation du M.A.G.* 91
3.3.5. *Le Diagramme de Flux de Données* 92

3.3.5.1. Les éléments du diagramme 92
3.3.5.2. Domaines d'application d'un D.F.D. . . 94
3.3.5.3. Avantages d'un D.F.D. 94
3.3.5.4. Inconvénients d'un D.F.D. 95
3.3.6. La Structure Hiérarchique des Activités et Tâches 96
3.3.6.1. Présentation de la S.H.A.T. 96
3.3.6.2. Domaine d'application d'une S.H.A.T. . 97
3.3.6.3. Avantages d'une S.H.A.T. 98
3.3.6.4. Inconvénients d'une S.H.A.T. 98
3.3.7. Le diagramme Input Process Output. 98
3.3.7.1. But d'un I.P.O. 99
3.3.7.2. Présentation d'un I.P.O. 99
3.3.7.3. Domaines d'application d'un I.P.O. . . 101
3.3.7.4. Avantages d'un I.P.O. 101
3.3.7.5. Inconvénients d'un I.P.O. 101
3.3.8. Le pseudo-code ou langage structuré 102
3.3.8.1. La syntaxe du langage 102
3.3.8.2. Domaine d'application du pseudo-code . 103
3.3.8.3. Avantages du pseudo-code 104
3.3.8.4. Inconvénients du pseudo-code 104
3.3.9. Les tables de décision 105
3.3.9.1. Présentation des tables de décision . 105
3.3.9.2. Domaines d'application des tables de décision 108
3.3.9.3. Avantages des tables de décision . . . 108
3.3.9.4. Inconvénients des tables de décision . 109
3.3.10. Jackson Structured Programming 110
3.3.10.1. Présentation succincte de la méthode 110
3.3.10.2. Domaines d'application de J.S.P. . . 111
3.3.10.3. Avantages de J.S.P. 112
3.3.10.4. Inconvénients de J.S.P. 112
3.4. Les outils d'aide au développement de logiciels 113
3.4.1. Le dictionnaire de données DataManager . . 113
3.4.1.1. Présentation de DataManager 114
a: Le langage de définition des données . . 114

b: Le langage de manipulation des données . 115
3.4.1.2. *Les avantages de DataManager* 116
3.4.1.3. *Les inconvénients de DataManager* . . . 117
3.4.2. *PALOMA* 118
3.4.2.1. *Les buts de PALOMA* 119
3.4.2.2. *Présentation de PALOMA* 119
a: Résultats fournis par PALOMA 119
b: Informations utilisées par PALOMA 120
3.4.2.3. *Avantages de PALOMA* 121
3.4.2.4. *Inconvénients de PALOMA* 122
3.4.3. *Les outils C.A.S.E.* 123
3.4.3.1. *EXCELERATOR* 125
a: Les possibilités d'EXCELERATOR 126
b: Les avantages d'EXCELERATOR 129
c: Les inconvénients d'EXCELERATOR 129
3.4.3.2. *TELON* 130
a: Présentation de TELON 131
b: Remarques concernant TELON 133
3.4.3.3. *Synthèse critique des outils C.A.S.E.* 134
3.5. *Classification des modèles et outils.* 136
3.5.1. *Le niveau conceptuel* 137
3.5.2. *Le niveau fonctionnel* 138
3.5.3. *Le niveau technique* 139
3.6. *Conclusion* 139

Chapitre 4 : Intégration de S.D.M. et des acquis méthodologiques de la C.G.E.R. 141
4.1. *Introduction* 142
4.2. *Restructuration proposée à la C.G.E.R.* 144
4.2.1. *Présentation de la démarche* 144
4.2.2. *Les nouvelles activités* 145
4.2.2.1. *Identification d'un problème* 145
4.2.2.2. *Demande de développement* 146
4.2.2.3. *Etude de l'existant* 147
4.2.2.4. *Exigences et limites du nouveau système* 149
4.2.2.5. *Modèle conceptuel des données* 150

4.2.2.6. Modèle conceptuel des traitements . . 150
4.2.2.7. Elaboration de solutions-comment développer ? 151
4.2.2.8. Choix d'une solution-qui développe ? . 152
4.2.2.9. Exigences et limites détaillées du nouveau S.I. 153
4.2.2.10. Décomposition en sous-systèmes . . . 154
4.2.2.11. Analyse des traitements 155
4.2.2.12. Analyse des données 156
4.2.2.13. Exigences en sécurité 156
4.2.2.14. Description des données 157
4.2.2.15. Description des programmes 157
4.2.2.16. Interface Homme - Machine 157
4.2.2.17. Plan de tests des programmes 158
4.2.2.18. Facilités requises ou imposées . . . 159
4.2.2.19. Tests du matériel 160
4.2.2.20. Codage 160
4.2.2.21. Tests 161
4.2.2.22. Préparation du test d'acceptation . . 161
4.2.2.23. Analyse de l'impact organisationnel . 162
4.2.2.24. Tests de formation 163
4.2.2.25. Test d'acceptation provisoire 163
4.2.2.26. Plan de conversion et de mise en production 164
4.2.2.27. Conversion et mise en production . . 164
4.2.2.28. Test d'acceptation définitif 165
4.2.2.29. Formation 165
4.2.2.30. Gestion et maintenance du système . . 166
4.2.2.31. Mort du système 166
4.3. Localisation des contrôles jalonnant le développement 166
4.3.1. Des contrôles : Pourquoi ? Par qui ? . . . 166
4.3.2. Présentation des points de contrôles . . . 169
4.3.2.1. (P1) Validation de la demande de développement 169
4.3.2.2. (P2) Evaluation de l'opportunité du projet 169

4.3.2.3. *(P3) Contrôle de la validité du schéma du S.I.* 170
4.3.2.4. *(P4) Cohérence de la solution* 171
4.3.2.5. *(P5) Terminaison de PALOMA* 171
4.3.2.6. *(P6) Evaluation de l'analyse fonctionnelle* 172
4.3.2.7. *(P7) La programmation peut-elle commencer* 172
4.3.2.8. *(P8) Le nouveau S.I. est-il prêt à être testé* 173
4.3.2.9. *(P9) Le nouveau système est-il accepté* 173
4.3.2.10. *(P10) La conversion peut-elle commencer* 174
4.3.2.11. *(P11) Contrôle de la formation* . . . 174
4.3.2.12. *(P12) Les problèmes matériels sont-ils résolus* 174
4.3.2.13. *(P13) Le système est-il acceptable* . 174
4.4. *Utilité et domaines d'application de la restructuration* 175

PARTIE II : APPLICATION DE LA METHODE PROPOSEE A UN CAS PRATIQUE : LA GESTION DU COURRIER 177

Chapitre 1 : Projet initial 178

Chapitre 2 : Etude d'opportunité et analyse conceptuelle 183
2.1. *Identification d'un problème* 184
2.2. *Demande de développement* 184
2.3. *Etude de l'existant* 187
2.4. *Exigences et limites* 188
2.5. *Le modèle conceptuel des données* 191
2.5.1. *Le schéma Entité - Association* 191
2.5.2. *La définition des entités* 193
2.5.3. *La définition des associations* 198
2.6. *Modèle conceptuel des traitements* 203
2.7. *Identification des possibilités de développement* 208

2.8. *Choix d'une solution (qui va développer ?)* . . . 210

Chapitre 3 : ANALYSE FONCTIONNELLE 211
3.1. *Exigences et limites détaillées* 212
3.1.1. *Objectifs vis-à-vis de la clientèle* 212
3.1.2. *Objectifs vis-à-vis des services de gestion* 213
3.1.3. *Objectifs vis-à-vis de l'informatique* . . . 214
3.1.4. *Objectif vis-à-vis de la direction* 215
3.1.5. *Contraintes générales* 215
3.1.6. *Limites du nouveau système* 216
3.2. *Découpe en sous-systèmes* 216
3.3. *Analyse des données* 216
3.3.1. *La normalisation* 216
3.3.2. *Le modèle logique des données* 219
3.3.2.1. *Conventions* 219
3.3.2.2. *Le schéma M.A.G.* 221
3.3.2.3. *Les types d'articles du M.A.G.* 222
3.3.2.4. *Les contraintes d'intégrité* 225
3.4. *Analyse des traitements* 225
3.5. *Exigences en sécurité* 236
3.6. *Préparation du test d'acceptation* 237
3.7. *Analyse de l'impact organisationnel* 238
3.8. *Le matériel à acquérir* 239

Chapitre 4 : Conclusion 241

<u>PARTIE III</u> : PRISE DE CONSCIENCE DES AVANTAGES ET DES LIMITES DES METHODES 243

Chapitre 1 : Synthèse critique de la méthode S.D.M. . . 244
1.1. *Introduction* 245
1.2. *Un cycle de vie d'un projet de développement [AVL]* 245
1.2.1. *L'analyse des besoins ou analyse d'opportunité* 245
1.2.2. *Spécification fonctionnelle du système informatisé* 245
1.2.3. *La conception* 246

1.2.4. Le codage 247
1.2.5. Conduite des tests, intégration et mise au point 247
1.2.6. La maintenance 248
1.3. Critique de la découpe en phases de S.D.M. . . . 248
1.4. Critique de la démarche méthodologique de S.D.M. 250
1.5. Synthèse des avantages et inconvénients de S.D.M. 252

Chapitre 2 : Conclusions 255

Liste des sigles utilisés 265
Bibliographie 267

Figures

Figure 2.1 : Planning de la phase 1 28
Figure 2.2 : Planning de la phase 2 35
Figure 2.3 : Planning de la phase 3 42
Figure 2.4 : Planning de la phase 4 46
Figure 2.5 : Planning de la phase 5 51
Figure 2.6 : Planning de la phase 6 55
Figure 2.7 : Planning de la phase 7 59

INTRODUCTION

A l'heure actuelle, l'un des sujets les plus controversés dans les entreprises est sans nul doute l'utilisation d'une méthodologie de développement standard pour les logiciels. Alors que la majorité déclare que les systèmes d'information causent beaucoup trop de problèmes au cours du développement et, surtout, lors de la maintenance, les avis sont plus disparates en ce qui concerne les causes de ces problèmes.

D'une part, les développeurs refusent les méthodes. Selon eux, elles ne sont pas réellement efficaces et causent généralement des pertes de temps. D'autre part, les méthodologistes argumentent que sans une approche rationnelle et contrôlée, il n'est pas possible d'élaborer des systèmes d'information aussi puissants que ceux requis par les organisations modernes.

Récemment, les outils logiciels d'aide à la conception des systèmes d'information ont ajouté une nouvelle dimension au problème. Leur but est d'améliorer la productivité des développeurs, mais l'absence de standards combinée à une utilisation anarchique a rendu ces logiciels beaucoup moins productifs qu'escompté.

Depuis plusieurs années, les méthodologistes de la C.G.E.R. tentaient de mettre au point une méthode maison pour le développement de logiciels. Ils étaient parvenus à standardiser le niveau conceptuel et une partie du niveau fonctionnel. L'utilisation de la programmation structurée était également proposée. Cependant, leur approche était nettement incomplète et, les besoins dans ce domaine étant de plus en plus pressants, il fut impératif pour eux de trouver une méthode capable de combler les vides existants. C'est principalement ce problème qui a motivé la réalisation de ce travail.

Lors du stage précédant la rédaction de ce mémoire, nous avons été amenés à examiner les possibilités de mise en place de la méthode choisie par la C.G.E.R. dans l'environnement partiellement défini existant. Les résultats auxquels nous sommes parvenus sont repris en détail dans ce travail. Remarquons que notre tâche n'a constitué qu'une infime partie de toute la recherche méthodologique nécessaire à l'obtention de résultats probants et que maints efforts restent à fournir.

Au cours de la première partie du mémoire, nous nous attarderons plus particulièrement sur les problèmes théoriques liés au développement des systèmes d'information.

Le premier chapitre sera consacré à la présentation de l'environnement de développement dans lequel nous avons évolué. Nous verrons pour quelles raisons il a dû être complété par l'ajout d'une nouvelle méthode et quels ont été les critères de choix de celle-ci.

Dans le second chapitre, nous présenterons la méthode qui a été choisie par les responsables en fonction des critères identifiés.

Les modèles et les outils sont, selon nous, indissociablement liés aux méthodes. Mais pourquoi sont-ils si souvent négligés lors de l'analyse ? Les développeurs invoquent leur lourdeur, leur inutilité. Ont-ils tort ? Dans le troisième chapitre nous tenterons d'apporter quelques éléments de réponse à ces deux questions en étudiant différents modèles et outils d'aide à la conception. Nous verrons comment ils pourraient être employés, quels sont leurs avantages, leurs limites mais aussi leurs dangers.

L'ultime chapitre de cette première partie aura pour but d'intégrer la méthode dans l'environnement contraignant en envisageant l'emploi des différents modèles et outils analysés.

Comme nous l'avons signalé, les développeurs qualifient souvent les méthodes d'inutiles. Notre manque d'expérience ne nous autorise pas à infirmer cette allégation. Néanmoins, nous avons pensé que l'utilisation rigoureuse d'une d'entre elles nous amènerait peut-être à mieux comprendre pourquoi elles sont mal acceptées et d'évaluer l'aide substantielle qu'elles pourraient apporter.

Dans cet objectif, la seconde partie de ce travail sera consacrée à la réalisation d'une application selon la méthode élaborée dans la première partie.

Enfin, dans la dernière partie, nous tenterons d'évaluer la méthode introduite et nous donnerons quelques réflexions au sujet d'une méthode "idéale".

PARTIE I

TENTATIVE DE MISE AU POINT D'UNE METHODOLOGIE DE DEVELOPPEMENT UTILISABLE DANS UN ENVIRONNEMENT CONTRAIGNANT (LA C.G.E.R.)

Introduction

Le but de cette première partie est d'arriver à une proposition de méthode pour le développement de logiciels applicable à la C.G.E.R., c'est-à-dire qui tienne compte des éléments méthodologiques déjà acquis dans cette maison.

Pour ce faire, nous allons dans un premier temps examiner l'environnement de développement existant. Nous identifierons, d'une part, ses points forts et, d'autre part, ses lacunes.

Ensuite, nous présenterons la méthode de développement choisie par la Caisse d'Epargne qui semble combler au mieux les déficiences actuelles.

Dans un troisième temps, nous mènerons une brève étude de certains modèles et outils d'aide au développement et nous tâcherons de découvrir comment ils pourraient être employés dans le cadre d'une méthode optimale.

Enfin, dans l'ultime chapitre de cette partie, nous tenterons de reconstituer, à partir des éléments contraignants exposés dans les chapitres précédents, une méthode qui soit acceptable et utilisable par la C.G.E.R., mais qui soit aussi aisément adaptable à d'autres environnements.

Chapitre 1

Présentation de l'environnement de développement de la C.G.E.R.

1.1. Introduction

Suite à l'introduction des S.G.B.D. (I.M.S.), la C.G.E.R. a essentiellement porté ses efforts méthodologiques sur l'analyse conceptuelle, et particulièrement le volet "données". Des résultats positifs ont été enregistrés dans ce sens et l'institution souhaite bien entendu conserver ses acquis. Simultanément, la programmation structurée a été encouragée et la méthode J.S.P. prônée. Pourtant, en ce qui concerne les autres stades du développement, la méthode reste muette.

Ces lacunes posent de graves problèmes en ce sens que, au moment d'aborder un stade non couvert par cette méthode, les développeurs[1] vont devoir travailler à l'expérience, d'où les résultats seront souvent d'une qualité très variable.

Le problème qui se posait donc à la C.G.E.R. était de conserver ses acquis et de les enrichir de manière à constituer pour les concepteurs une guidance tout au long du développement de manière à obtenir des logiciels plus homogènes et de meilleure qualité.

1.2. Distinction préalable : cycle de vie - méthode

Afin d'éviter toute confusion, nous allons préciser dès maintenant ce qu'est le cycle de vie d'un projet[2].

A notre avis, indépendamment de toute méthode, le cycle de vie d'un projet informatique se compose des différents stades successifs par lesquels il transitera tout au long

---

1. Les développeurs (ou concepteurs) sont les utilisateurs et les informaticiens responsables du développement d'un projet.

2. Nous présenterons un cycle de vie qui nous semble idéal dans le premier chapitre de la troisième partie.

de son existence. Le degré de précision avec lequel chacun de ces stades est traité variera souvent en fonction du type de projet mais aussi de l'environnement de développement.

Par exemple, un projet passera toujours par un stade de définition au cours duquel les exigences le concernant seront, formellement ou non, précisées. Plus tard dans le développement, le projet en sera au stade de la programmation.

Cependant, des différences dans ce cycle de vie pourront apparaître au travers des différentes méthodes. Elles l'adapteront souvent selon une optique particulière. Certaines insisteront davantage sur certains stades qu'elles considèrent comme primordiaux alors qu'elles en délaisseront d'autres. Une méthode pourra également présenter un stade qui n'a jamais été abordé par d'autres méthodes. On pourra alors considérer que cette méthode a ajouté une nouvelle étape au cycle de vie.

Le cycle de vie a donc une existence autonome qui peut, à l'occasion, faire l'objet de modifications et/ou d'ajouts pouvant le transformer considérablement.

1.3. L'environnement de développement de la C.G.E.R.

Le développement d'un système informatique vu par le groupe méthode du service M.A.I. (Méthodologie et Administration de l'Information) se compose de six étapes principales (l'analyse conceptuelle, l'analyse fonctionnelle, l'analyse technique, la programmation, les tests et le passage en production) et comporte deux volets (les données et les traitements).

Cette méthode se place dans la philosophie de la plupart des méthodologies de développement, à savoir la découpe du cycle de vie d'un projet en étapes bien définies. Elle

est principalement basée sur les données, et ce vu leur plus grande stabilité dans le temps.

Mais, comme nous allons le découvrir ci-après, cette "méthode maison" comporte plusieurs "trous" et n'offre donc pas un support continu. Ce manque de standardisation de certains stades du développement peut souvent être à la base d'un travail dont la qualité varie selon l'expérience du développeur et empêche un contrôle total de l'état d'avancement des projets.

Les lacunes méthodologiques de la Caisse d'Epargne ont été mises en avant par suite de problèmes internes sur lesquels nous ne nous attarderons pas. Ce qu'il nous semble quand même utile de signaler est que ce sont ces problèmes qui ont accéléré les recherches de standardisation et ont amené à intégrer les acquis dans le cadre plus général d'une nouvelle méthode.

La suite de ce chapitre est consacrée à l'exposé des différents stades du développement existants à la C.G.E.R. tels que nous les avons connus lors de notre passage au M.A.I.

1.4. Aperçu du développement des projets informatiques

Dans cette section, nous nous bornerons à la présentation des différentes étapes du développement d'un système à la C.G.E.R. Les outils et modèles éventuellement cités feront l'objet d'une étude approfondie au chapitre trois ci-dessous.

1.4.1. La demande de développement

La demande de développement précède tout développement de projet et permet, sur base des renseignements fournis par l'utilisateur[3], de donner le feu vert pour démarrer l'analyse.

Elle doit être approuvée par le D.S.I. (Développement des Systèmes d'Information) et est transmise pour information au M.A.I. (section "administration de l'information").

Cette demande est utilisée à des fins multiples (demande d'un nouveau système, demande d'une extension ou d'une modification conceptuelle, demande de nouveau matériel) et son rôle est, dès lors, très variable.

Soit elle permettra d'évaluer la priorité à accorder au projet ou sa faisabilité en fonction de la charge actuelle et du nombre de demandes en suspens (dans le cas d'une demande de nouveau matériel, y a-t-il un budget suffisant ?), soit elle sera purement administrative (dans le cas où le développement d'un système est décidé par la direction suite aux nécessités économiques).

Un exemplaire de cette demande de développement est repris en annexe A.

1.4.2. La définition de projet

La définition de projet précise les exigences (objectifs, besoins) et les limites (frontières) du système à développer, et ce, sur base du système

---

3. L'utilisateur est celui qui travaille avec le système existant et qui travaillera avec le système futur.

existant. Elle comporte trois volets : l'étude de l'existant, les objectifs et contraintes du nouveau système et les solutions envisageables.

Sur base de ce dossier -établi par les membres du groupe d'étude[4], la hiérarchie[5] décidera quelle solution choisir parmi les différentes propositions.

1.4.3. L'analyse conceptuelle

Le but de l'analyse conceptuelle est de produire des modèles représentatifs du réel. Elle "se situe à un niveau d'invariance suffisant pour garantir à long terme un maximum de stabilité aux bases du nouveau système.

Ce niveau d'abstraction lui interdit strictement de prendre en compte des contingences techniques ou organisationnelles" [MAI 1, p.1].

Au cours de cette analyse, "l'accent sera principalement mis sur la partie 'données', et ce pour deux raisons :

- D'une part, la plus grande stabilité des données. Deux systèmes de même finalité, dans deux entreprises différentes, manipuleront en général des informations fort proches. L'ajout d'un nouveau type fondamental de donnée est en outre un événement relativement rare dans la vie d'un système.

---

4. Le groupe d'étude est composé d'une part d'informaticiens délégués par le D.S.I., d'autre part d'utilisateurs sensés apporter un maximum d'aide et de connaissance pour le développement.

5. La hiérarchie est la désignation globale des personnes habilitées à prendre une décision finale dans un domaine défini.

- D'autre part, le modèle conceptuel des données servira ultérieurement à l'élaboration des architectures des supports d'information, et, en particulier des bases de données" [MAI 2, p.2].

"Les traitements, par contre, répondent à des besoins ponctuels propres à chaque organisation et ne constituent donc pas la partie stable d'un système. Il convient donc de les étudier séparément des données, afin de pouvoir ultérieurement, lors de l'étape PALOMA[6], valider l'un par l'autre les modèles conceptuels des données et des traitements" [MAI 3, p.4].

1.4.3.1. Le volet "données"

Du côté des données, utilisateurs responsables de la définition et du fonctionnement du nouveau système et informaticiens en charge du projet vont concevoir ensemble un modèle conceptuel des données de type Entités - Associations. Ce schéma sera complété par l'adjonction des volumes estimés.

Le choix de cette technique de modélisation a été motivé par le fait qu'"elle constitue (...) un excellent moyen d'aborder un problème, ainsi qu'un outil privilégié de dialogue entre utilisateurs et informaticiens" [MAI 2, p.3].

Le modèle élaboré sera soumis au M.A.I. pour approbation.

---

6. Cfr. infra 1.4.4. et 3.4.2.

1.4.3.2. Le volet "traitements"

Précisons tout d'abord que l'introduction de modèles au niveau de l'analyse conceptuelle des traitements est ultérieure à celle faite au niveau des données.

La modélisation conceptuelle des traitements repose sur plusieurs modèles établis par le groupe d'étude sur base des renseignements fournis par l'utilisateur.

Tout d'abord, la décomposition des traitements va permettre, grâce à une approche "top-down", de réduire le problème initial en sous-problèmes de complexité inférieure, et récursivement.

Ensuite, le diagramme des flux de données identifiera les différents flux d'informations circulant au travers du système et transformés par les traitements successifs[7].

Enfin, le modèle de description des tâches, via un diagramme I.P.O. (Input Process Output), va définir chaque tâche en fonction de ses entrées et de ses sorties.

Un modèle supplémentaire, le modèle de la dynamique, décrira quant à lui les "processes", "enchaînements de plusieurs tâches faisant partie éventuellement d'activités différentes correspondant à un problème de base de l'utilisateur" [DEDE, p.19].

---

7. Une DONNEE est une "représentation (codée) des propriétés -y compris l'existence- d'un concept, d'un objet, d'un fait ou d'un événement." Une INFORMATION est la "signification potentielle attachée aux données, susceptible d'affecter le comportement des hommes et des machines dans une organisation" [BOPI, II.c].

Notons pour terminer que l'utilisation du dictionnaire de données DataManager est indispensable pour pouvoir exécuter l'étape PALOMA ci-dessous. On introduira notamment dans ce dictionnaire la description des données et des traitements.

1.4.4. PALOMA

PALOMA (PAth LOad MAtrix) est une application développée à la C.G.E.R. destinée à fournir une appréciation brute des accès aux données du nouveau système.

"PALOMA se base sur le contenu du dictionnaire de données DataManager et intègre les résultats de l'analyse des données et de l'analyse des traitements.

PALOMA :

- valide le modèle conceptuel des données;
- permet d'établir le modèle dynamique des données, étape essentielle dans l'architecture des bases de données;
- estime la charge représentée par les tâches du système" [MAI 1, p.3].

La validation effectuée par PALOMA se situe à deux niveaux distincts.

Tout d'abord, le fait d'introduire le schéma conceptuel constitue en soit une validation.

Ensuite, une validation plus importante du schéma lui--même va avoir lieu, d'une part, via les estimations de trafic inter-objets (entités ou associations porteuses). En effet, si un trafic important entre deux objets est rapporté par PALOMA et qu'aucune relation n'avait été prévue dans le schéma initial, il sera peut-être

nécessaire de revoir ce schéma en ajoutant par exemple une nouvelle association entre les deux objets concernés. D'autre part, l'exécution de PALOMA permettra de s'assurer que toutes les informations utilisées par les tâches sont bien présentes dans le modèle.

PALOMA est orienté données (validation du modèle conceptuel des données par confrontation avec le modèle conceptuel des traitements). Cette approche n'est pas le fruit du hasard. Selon nous, elle est influencée par le fait que la C.G.E.R. dispose d'un matériel et de logiciels précis et compétitifs (matériel I.B.M., gestionnaire de base de données I.M.S. et D.B.2) qui leur permettent d'avoir une telle approche.

1.4.5. L'analyse fonctionnelle

"Il s'agit ici de fournir l'information nécessaire à l'architecture des programmes et des structures de données (en particulier des bases de données) et de préciser la manière dont l'utilisateur veut voir fonctionner son système" [MAI 1, p.3].

La scission données - traitements reste présente à ce niveau.

1.4.5.1. Le volet "données"

Si nécessaire, et en tout cas lors de l'utilisation d'un S.G.B.D. (Système de Gestion de Base de Données) relationnel, un processus de normalisation sera appliqué.

Ensuite, et en tenant compte des résultats fournis par PALOMA, on transformera le modèle conceptuel des données en un schéma logique des données. Cette transformation sera validée par le "Data Base

Administrator" (Le D.B.A. est la personne chargée de développer, d'entretenir et de convertir les bases de données. Il construira les modèles logiques et physiques des données en fonction des S.G.B.D. utilisés).

1.4.5.2. Le volet "traitements"

Du côté des traitements, une distinction supplémentaire doit être faite entre le batch et l'interactif.

En ce qui concerne les tâches batch, on définira la logique d'enchaînement des traitements (on spécifie ces traitements ici) et on décrira les états de sortie (imprimés,...).

Pour les tâches interactives, on spécifiera les dialogues utilisateurs - système et on élaborera les écrans qui supporteront ces dialogues.

Au cours de cette analyse, on sera également amené à identifier les transactions utilisateur et les transactions base de données.

"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" [MAI 4, 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" [DEDE, p.20].

1.4.6. L'analyse technique

C'est au niveau de l'analyse technique que les besoins méthodologiques prennent le plus d'ampleur à la Caisse d'Epargne. En effet, le "comment faire" est indéfini et aucune technique précise n'est préconisée. Seule l'expérience des développeurs permet donc d'aboutir aux spécifications détaillées du système.

Au niveau des données, le modèle logique issu de l'analyse fonctionnelle va être transformé en un schéma physique optimal respectant les contraintes du S.G.B.D. utilisé. Ce travail est réalisé par le D.B.A..

Au niveau des traitements, les analystes du C.T.I. (Centre de Traitement de l'Information) procéderont à la description complète des programmes et de leurs enchaînements.

1.4.7. La programmation et les tests

La méthode de Jackson, au même titre que les tables de décision et que le langage structuré, est fortement recommandée pour les phases de programmation et de test.

Les programmeurs sont responsables de la correction de leurs programmes et ils sont également en charge des tests d'intégration de ceux-ci.

Les utilisateurs, quant à eux, sont représentés par des testeurs chargés de la mise au point et de l'exécution des tests d'acceptation. Le but de ces tests est de vérifier si le système répond bien aux desiderata initiaux des utilisateurs ainsi qu'à toutes les exigences fonctionnelles définies au cours de l'analyse.

Pour que ces tests soient les plus fructueux possible, il convient que les testeurs élaborent des procédures de tests, qu'ils créent l'environnement nécessaire à leurs exécutions (par exemple les données qui les supporteront) et qu'ils prévoient, enfin, les résultats attendus pour chacun d'eux.

Signalons qu'il existe à la Caisse d'Epargne un document [MAI 8] précisant comment répartir le travail entre analystes et programmeurs pour réaliser les programmes. Ces deux types d'acteurs doivent collaborer étroitement pour utiliser J.S.P.

1.4.8. Le passage en production

Lors du développement, le programme se trouve dans une librairie de développement. Il est ensuite transféré dans un environnement particulier où il subira les différents tests. Enfin, quand il est accepté, il est transféré dans l'environnement de production où il devient opérationnel.

Remarquons qu'au moment du transfert en production, un minimum de documentation (noms des fichiers, caractéristiques) est fourni aux opérateurs pour leur permettre de lancer le nouveau système.

1.5. Conclusion

Comme nous venons de le voir, la méthodologie de développement proposée à la C.G.E.R. comporte des éléments très précis mais reste parfois ambiguë, notamment en ce qui concerne les traitements. Cette faiblesse du volet "traitements" vient du fait que les recherches sont principalement basées sur les données à cause de leur plus grande stabilité dans le temps.

Certains stades du développement sont particulièrement bien établis (PALOMA par exemple) et ne peuvent en aucun cas être supprimés par l'arrivée d'une nouvelle méthodologie de développement.

Le schéma repris en annexe B montre bien la dégradation de la méthode au cours du développement. Les lacunes apparaissent particulièrement aux niveaux de l'analyse fonctionnelle des traitements et de l'analyse technique.

Le problème qui se posait donc au M.A.I. était de trouver une méthode qui soit suffisamment souple pour permettre de garder les acquis méthodologiques, mais qui soit également garante de plus de rigueur tout au long du développement.

Chapitre 2

Présentation de la méthode S.D.M.

2.1. Remarque préliminaire

Le but de ce chapitre est de présenter la méthode S.D.M. -System Development Methodology- et non de la justifier. Son évaluation sera réalisée dans le premier point de la troisième partie. Cette présentation s'inspire essentiellement de la version néerlandaise de la méthode [SDM].

Dans un premier temps, nous essayerons de déterminer les différents concepts de la méthode. Ensuite, nous exposerons brièvement les phases et les activités qui la composent. Pour les différentes phases, nous présenterons un planning possible des différentes activités, proposé par [SDM]. Il est clair, cependant, que plusieurs plannings existent pour une même phase, suivant le projet à développer et l'organisation de l'entreprise. Dans celui adopté par [SDM], une activité figure dans le planning si elle ne peut plus être reportée. Les activités restantes en sont dépendantes. A la fin de ce chapitre, nous donnerons une vue globale sur l'évolution des données, des traitements et de la sécurité dans la méthode.

2.2. La méthode S.D.M.

La méthode S.D.M. -que nous désignerons par la suite par SDM- traite du cycle de vie complet d'un système d'information[8]. Elle s'occupe de l'analyse, de la réalisation et de la maintenance du projet.

SDM se veut un outil répondant aux besoins :
- de gérer, de manière uniforme, les projets se présentant dans une entreprise;
- de constituer la documentation d'un système;

---

8. Dans la partie consacrée à la critique de SDM, nous présenterons un cycle de vie d'un système d'information et nous examinerons comment SDM le traite.

- de développer et gérer correctement des systèmes d'information.

SDM n'impose pas l'utilisation de techniques particulières d'analyse et/ou de réalisation. Ce choix est laissé à l'appréciation de l'entreprise qui utilise SDM. En effet, elle décrit le QUOI faire et non le COMMENT faire. Toutefois, il n'est pas toujours possible d'expliquer le QUOI sans dire le COMMENT car ces deux composantes ne sont pas totalement indépendantes. En général, SDM essaye de bien séparer le deux.

> Exemple : Pour les données au niveau conceptuel, le QUOI faire, c'est concevoir la structure des données (rassembler les données, les grouper, regarder leurs relations, ...), le COMMENT faire, c'est réaliser un schéma entité/relation.

En complément, des outils et des modèles sont proposés pour supporter différentes activités et ainsi en faciliter la réalisation.

SDM analyse le cycle de vie d'un système en le décomposant en phases et en décrivant, pour chacune d'elles, des activités. La méthodologie "phasée" permet, notamment, l'évaluation de ce qui est réalisé et de ce qui reste à faire. De plus, en cas de modification, elle donne le moyen de savoir où retourner et quelles activités changer.

Dans SDM, les activités à réaliser lors du développement sont classées en 7 phases :

0. Etude préliminaire
1. Etude de l'existant et Analyse conceptuelle
2. Analyse fonctionnelle
3. Analyse technique
4. Réalisation des programmes
5. Tests
6. Conversion et Mise en production
7. Utilisation et Gestion du système

Remarques :

- La phase 0 -l'étude préliminaire- ne fait pas partie intégrante de SDM[9]. En effet, SDM s'intéresse au développement d'un système d'information. Or cette réalisation suppose qu'un problème est apparu et qu'il a été décidé de le solutionner. Cette détection et cette décision s'effectuent au cours de la phase 0. SDM commence, donc, l'analyse d'un problème à la fin de cette phase. En bref, on peut dire que les activités qui la composent consistent à cerner les problèmes qui se présentent sur le plan de la production d'informations, à fixer les limites dans lesquelles il faut effectuer une étude, à déterminer les priorités des demandes de développement.

- Les phases 1 et 2 de SDM, c'est-à-dire l'étude de l'existant, l'analyse conceptuelle et l'analyse fonctionnelle, sont réalisées pour le système dans son ensemble et s'occupent de la conception du système.

- Les phases 2 à 6 peuvent se dérouler pour le système entier ou par sous-systèmes et s'occupent de la réalisation. Il est possible de développer les sous-systèmes individuellement.

- Pour de petits projets, certaines phases sont combinées (car le processus de développement est très simple). Lors de la fusion de phases, SDM ne cite pas les activités à réaliser.

En bref, on peut dire que le découpage en phases sert :
- à déterminer les différentes étapes du cycle de vie d'un système, et

---

9. Dans la version anglaise de la méthode [SDM-ANG], cette phase fait partie de la méthode.

- à diviser le développement du système en unités faciles à manier et à appréhender. (Remarque : certaines phases peuvent se chevaucher)

Cela rend possible :
- une diminution des risques de retour en arrière,
- un juste emploi des compétences du personnel prévu pour le projet informatique (remarque : les réalisateurs des différentes activités ne sont pas identifiés. Mais l'énoncé du contenu de celles-ci permet de déduire les catégories de personnes utiles à leur exécution.),
- une planification des phases du développement,
- une prise de décision sur base du rapport fourni après chaque phase (pour le démarrage de la phase suivante).

Les activités citées constituent une "checklist" générale. Pour certains projets, une partie de celles-ci peut être omise (car pas d'application ou banale). Toutefois, il est souhaitable d'en justifier la raison pour compléter la documentation du projet et pour signaler que toutes les activités ont été abordées.

En effet, cette liste peut servir à l'élaboration d'un échéancier pour le projet. A tout moment, on pourra contrôler que les activités, qui devaient être menées, ont été effectivement abordées et exécutées. Pour contrôler la progression, le coût, la planification d'un projet, des rapports d'activités réguliers sont nécessaires. SDM offre une aide pour constituer une documentation complète, claire, précise, ... Tout en développant le système, on en constitue la documentation. Le contenu de la documentation de chaque activité est, en général, décrit.

> Exemple : Pour l'activité 1.6 -Faire un schéma du système-, la documentation doit contenir [DOL]:
>
> - la description des entrées/sorties;
> - les relations entre entrées et sorties;
> - les données et les groupes de données;
> - la structure des données;
> - les fonctions du nouveau système;
> - une vue d'ensemble des traitements principaux.

La documentation, ainsi élaborée, peut faciliter la communication entre toutes les parties concernées par le projet. En effet, si les personnes intervenant de près ou de loin dans le développement du système d'information connaissent la méthode, les rapports des activités sont une base de discussion entre les développeurs et la direction (points de décision, indications sur l'avancement du projet), les utilisateurs (vérification de conformité, d'avancement), surtout aux points d'évaluation (c'est-à-dire à la fin des phases).

SDM se veut "orientée vers les utilisateurs". Elle s'appuie sur une forte participation de ceux-ci lors du développement du système. De plus, elle traite très tôt les problèmes humains (dialogue homme/machine, relation homme/machine, procédures manuelles).

Pour le bon déroulement du projet, une personne unique devrait servir d'intermédiaire entre les utilisateurs et les développeurs. Cette personne, "le correspondant informatique", aurait une connaissance approfondie de l'organisation pour être reconnu par les utilisateurs et une connaissance étendue de l'informatique (possibilité et faisabilité). De plus, elle aura une aptitude aux contacts car elle devra concilier les parties en présence et modérer leurs exigences. D'autre part, elle centralise les demandes des utilisateurs, les réponses des développeurs pour garder la cohérence du système.

Enfin, SDM donne aussi la manière de conduire chaque activité.

> Exemple : Pour l'activité 1.1 -Cerner le problème et le champ de l'étude de l'existant et de l'analyse conceptuelle-, il faut procéder de la manière suivante [SDM]:
>
> 1) interviewez ceux qui ont détecté le problème au départ et formulez avec eux la définition du problème;
> 2) analysez le problème plus profondément pour en connaître clairement l'origine et quelles en sont les conséquences directes;

3) étendez l'analyse si nécessaire jusqu'aux problèmes apparentés;
4) répétez les trois premières étapes tant que l'on n'est pas arrivé à une description du problème acceptée par tous;
5) rédigez les causes et les limites du problème;
6) incorporez le but final de l'étude (quels problèmes seront solutionnés);
7) faites des évaluations du temps et des frais à l'égard des activités à faire, et comment et par qui, elles peuvent être exécutées;
8) déterminez en conseil la division du travail dans le groupe du projet, ses compétences et ses responsabilités;
9) déterminez quelles méthodes spéciales et outils doivent éventuellement être utilisés (utiliser SDM, tables de décision, ...);
10) faites un plan d'activités avec différents points d'évaluation;
11) déterminez dans le groupe ce que doit être le contenu du rapport final;
12) consignez les résultats des étapes précédentes dans un rapport et présentez ceux-ci aux décideurs;
13) répétez si nécessaire les étapes précédentes pour obtenir un consentement unanime;
14) placez tous les documents dans le dossier du projet.

SDM constitue donc un support pour la conduite de projet et pour la documentation.

2.3. Phase 1 : Etude de l'existant et Analyse conceptuelle

2.3.1. Description

Son but est de comprendre le domaine à étudier et d'en fixer les limites. Au cours de cette phase, la définition du problème est précisée, les processus existants sont étudiés.

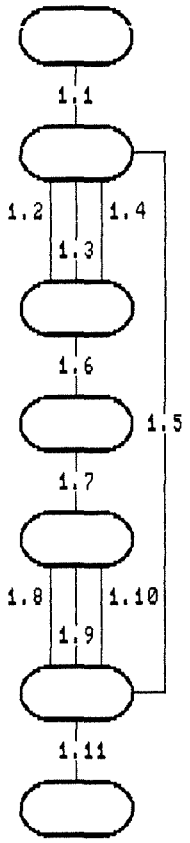
Ainsi, il convient :
- de fixer les objectifs justifiant la poursuite du développement,
- d'envisager différentes conceptions possibles du système,
- d'effectuer une analyse bénéfices/coûts globale.

La documentation résultante décrit les contraintes et les critères fonctionnels auxquels devra satisfaire le système. Elle sert de base pour la poursuite du développement. Ce rapport donne aux réalisateurs une mine de renseignements pour la suite du projet et la maintenance (sur les coûts, le temps de réalisation, le matériel informatique nécessaire, le personnel à utiliser, ...). Il permet aux décideurs de choisir en fonction de l'approche bénéfices/coûts s'il y a lieu de poursuivre le développement d'un système capable d'atteindre les objectifs fixés, de refaire une analyse. Il peut aussi servir de base pour un cahier des charges.

Les responsables de la phase sont les utilisateurs.

En résumé, l'analyse conceptuelle :
- doit prouver l'utilité d'un nouveau système et donner une analyse bénéfices/coûts;
- doit constituer la base de la prise de décision de démarrer un projet (conception et réalisation);
- sert à produire une bonne définition du problème ainsi qu'un plan d'approche et une liste d'exigences.

2.3.2. Planning possible des activités



Activité 1.1: Cerner le problème et le champ de l'étude de l'existant et de l'analyse conceptuelle

Activité 1.2: Rassembler les données sur la situation existante

Activité 1.3: Les analyser et les évaluer

Activité 1.4: Déterminer les objectifs et exigences du nouveau système

Activité 1.5: Arrêter les points qui restent à résoudre et les hypothèses de base

Activité 1.6: Faire un schéma du système

Activité 1.7: Déterminer les outils et les solutions possibles

Activité 1.8: Evaluer les solutions et opérer une sélection

Activité 1.9: Déterminer les problèmes de conversion et de mise en production

Activité 1.10: Elaborer un planning global et une vue d'ensemble des coûts et bénéfices

Activité 1.11: Rédiger le rapport

Remarque : Les activités 1.2 à 1.4 comprises et 1.8 à 1.10 comprises forment souvent un processus itératif.

Figure 2.1 : Planning de la phase 1

2.3.3. Description des activités

Activité 1.1: Cerner le problème et le champ de l'analyse (objectifs, limites, exigences, planning des activités et des ressources humaines, coût, fréquence des

rapports, ...) de manière globale. C'est ici que le groupe d'étude[10] est constitué. Sur base du rapport de cette activité, une décision, sur la poursuite du développement, est prise. Si celui-ci se poursuit, toutes les informations recueillies seront détaillées dans les activités 1.2 à 1.4.

Informations utiles pour son exécution :
- le rapport de l'étude préliminaire (s'il existe);
- les manuels d'utilisation de l'ancien système;
- les directives, les normes de l'entreprise;
- la politique sociale de l'entreprise.

Aide possible[11] : un questionnaire.

Activité 1.2: Rassembler les données, susceptibles d'influencer de près ou de loin le nouveau système, sur la situation existante. Grâce à des interviews et des études de documents, on obtient un inventaire complet des informations sur l'organisation existante, le processus de production et le système d'information, nécessaires à la réalisation du système. La vue d'ensemble de l'entreprise et du processus de production permet l'émergence de problèmes (à éliminer si possible).

Informations utiles :
- la définition du projet (1.1);
- les rapports d'analyses précédentes de l'entreprise (sur l'organisation, la production);
- des schémas et descriptions du système actuel;

---

10. Ici, la notion de "groupe d'étude" regroupe les utilisateurs et informaticiens en charge du projet, responsables de la définition et du fonctionnement du nouveau système.

11. Ici, nous nous contentons uniquement de citer des outils et modèles possibles. Dans le prochain chapitre, nous donnerons des indications permettant de comprendre l'intérêt de ceux-ci dans les activités. Cette liste d'aides n'est bien évidemment pas exhaustive.

- les données, sur le système actuel, du dictionnaire (s'il existe).

Aides possibles :
- un questionnaire;
- des techniques d'interview;
- la statique des traitements (schémas IPO);
- des diagrammes de flux de données;
- des tables de décision;
- un dictionnaire de données.

Activité 1.3: Les analyser et les évaluer. On a une vue d'ensemble sur le système existant ainsi que ses points forts (à respecter), ses points faibles (à éliminer) et des quantifications. On entrevoit ici aussi les problèmes de sécurité et de confidentialité.

Informations utiles : Cfr. activité 1.2.

Aides possibles : Cfr. activité 1.2.

Activité 1.4: Déterminer les objectifs et exigences du nouveau système, les critères de prestation (temps de réponse), les contraintes organisationnelles (formation) et d'intégrité (sécurité). Les exigences sont classifiées car elles seront résolues à des stades différents du développement.

Informations utiles :
- la définition du problème, le champ de l'étude (1.1);
- l'analyse de l'existant, les souhaits et exigences, les suggestions pour améliorer le système (1.3);
- des informations techniques issues de la littérature scientifique ou de l'expérience d'une autre entreprise.

Activité 1.5: Arrêter les points qui restent à résoudre (mémento) et les hypothèses de base (limites imposées au développement en temps, en moyens, ...), les moyens pour développer le système (personnel, hardware,

software), les postulats (exclure des cas très exceptionnels).

Informations utiles :
- l'analyse de l'existant (1.3);
- les objectifs et exigences du système (1.4);
- les manuels internes;
- la politique sociale de l'entreprise;
- la législation.

Activité 1.6: Faire un schéma du nouveau système. On détermine, de manière globale, les entrées et sorties, les données, les traitements principaux, les fonctions du nouveau système, les besoins en information.

Informations utiles :
- la situation existante (1.2);
- l'analyse de l'existant (1.3);
- les objectifs et exigences du système, les critères de prestation (1.4);
- les limites et hypothèses (1.5).

Aides possibles :
- la modélisation conceptuelle des données;
- la structuration et la statique des traitements ;
- les diagrammes de flux de données;
- le dictionnaire de données;
- la normalisation.

Activité 1.7: Déterminer les outils (estimation du hardware et du software nécessaires) et les solutions possibles avec avantages et inconvénients (développement d'un prototype, package sur le marché).

Informations utiles :
- les objectifs du système d'information, les critères de prestation (1.4);
- les limites et utilitaires (1.5);
- les fonctions du système, la description des entrées et des sorties (1.6);

- d'autres systèmes et expériences de développement;
- des packages d'application disponibles sur le marché.

Activité 1.8: Evaluer les solutions et opérer une sélection. On décrit une procédure d'évaluation en fonction des coûts, de la confidentialité, de la flexibilité, de l'ergonomie, de la continuité, des conséquences organisationnelles, ...

Informations utiles :
- les buts et limites de l'étude (1.1);
- l'analyse de l'existant (1.3);
- les objectifs et exigences du système (1.4);
- les limites, hypothèses et utilitaires (1.5);
- la description des entrées/sorties et des fonctions du système (1.6);
- les solutions alternatives (1.7).

Activité 1.9: Déterminer les problèmes de conversion et de mise en production (acquisition du hardware et du software supplémentaires, problèmes de formation, déterminer les critères d'acceptation).

Informations utiles :
- la description du problème, les références (1.1);
- l'analyse de l'existant (1.3);
- les objectifs du système (1.4);
- la description des entrées/sorties, des utilitaires (1.5);
- la solution sélectionnée (1.8);
- les possibilités de formation.

Activité 1.10: Elaborer un planning global (pour toutes les phases) et une vue d'ensemble des coûts (uniques, répétés) et bénéfices (chiffrables, non chiffrables).

Informations utiles :
- les résultats des différentes activités;
- la politique budgétaire;
- l'analyse bénéfices/coûts.

Activité 1.11: Rédiger le rapport. Il est constitué des différents rapports d'activités et sert de base pour la prise de décision. (Remarque : il peut être rédigé en parallèle avec les autres activités).

Informations utiles : les résultats des activités.

Aide possible : un dictionnaire de données.

2.4. Phase 2 : Analyse fonctionnelle

2.4.1. Description

Lors de l'analyse fonctionnelle, c'est toujours ce que le système doit faire (LE QUOI) qui est étudié. Les fonctionnalités et les exigences du système choisi lors de l'étape précédente sont détaillées, l'analyse conceptuelle n'ayant pas pour but de décrire le système de manière approfondie.

Toutes les fonctions que le système devra remplir sont définies et, le cas échéant, décrites. Ces spécifications consistent en une analyse fonctionnelle de l'ensemble du système, y compris le matériel et le logiciel nécessaires, jusqu'au niveau où les programmes et les procédures manuelles peuvent être conçues et où le matériel peut éventuellement être commandé.

Tous les critères relatifs aux données à utiliser doivent être étudiés afin de définir la structure logique des données et de déterminer les "clés" par lesquelles on souhaite accéder à ces données. Les critères auxquels doivent répondre le logiciel et le matériel seront étudiés en détail et en corrélation avec ceux du système.

C'est au cours de cette phase que les sous-systèmes éventuels sont définis. L'analyse fonctionnelle peut se dérouler en deux passes. Lors de la première, le sys-

tème est développé en gros jusqu'à un niveau uniforme pour les sous-systèmes (pouvoir définir les interfaces). En effet, il arrive fréquemment qu'une partie intéressante soit développée en détail. Et, plus tard, lorsque l'on s'occupe d'une autre partie, il peut s'avérer qu'il y a lieu de modifier la partie développée antérieurement, ce qui signifie un gaspillage de temps et d'argent. Lors de la seconde, on développe en détail les sous-systèmes. Les interfaces et les relations entre celles-ci sont clairement définies. Une architecture de développement est élaborée. On précise l'ordre de réalisation des différents sous-systèmes (développement en parallèle, en série).

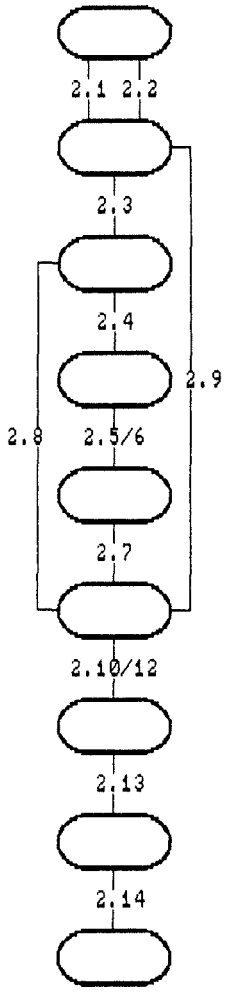
Il peut s'écouler un grand laps de temps entre la phase d'analyse conceptuelle et la phase d'analyse fonctionnelle. Il peut se produire des événements modifiant la solution choisie. Il est donc possible de retravailler l'analyse conceptuelle lors des premières activités de cette phase.

A la fin de cette phase, on a une conception claire et "définitive" du système global ainsi que les conditions auxquelles les sous-systèmes devront répondre. L'analyse fonctionnelle doit être détaillée suffisamment sans toutefois verser dans la technicité de sorte que l'utilisateur comprenne clairement les potentialités du système. En d'autres mots, la description de ce que le système doit faire est suffisamment précise pour qu'il ne subsiste plus d'ambiguïté pour le développement ultérieur. Mais celle-ci ne contiendra pas ou peu de notions techniques car les utilisateurs désirent connaître ce que le système pourra faire et non comment cela sera réalisé. De plus, cela facilitera la prise de décision de ceux-ci car ils ne sont pas nécessairement accoutumés aux notions techniques.

Les utilisateurs et les informaticiens sont responsables de cette phase. Les utilisateurs contrôlent que la

description reprend toutes les fonctionnalités souhaitées. Les informaticiens regardent si c'est techniquement possible et donnent une formulation claire et précise des spécifications.

2.4.2. Planning possible des activités



Activité 2.1: Spécifier les exigences du système y compris les exigences futures

Activité 2.2: Déterminer le cadre dans lequel le nouveau système devra fonctionner

Activité 2.3: Diviser le système en sous-systèmes et les décrire

Activité 2.4: Définir l'input et l'output par sous-système et les interfaces

Activité 2.5/6: Elaborer les diagrammes des traitements et les descriptions des activités

Activité 2.7: Spécifier les exigences en sécurité et confidentialité

Activité 2.8: Déterminer les problèmes humains et des solutions

Activité 2.9: Concevoir la structure logique des données

Activité 2.10/12: Spécifier les facilités requises en communication de données, en hardware et en software

Activité 2.13: Elaborer un plan pour la poursuite du développement et la mise en production (révision)

Activité 2.14: Rédiger le rapport d'analyse fonctionnelle

Figure 2.2 : Planning de la phase 2

2.4.3. Description des activités

Activité 2.1: Spécifier les exigences du système, y compris les exigences futures (croissance du système) pour garantir un niveau élevé de stabilité (conception modulaire, extensibilité hardware, intégration de surcapacité dès le départ, ...). Avant de démarrer l'analyse fonctionnelle, on contrôle que les exigences du système définies au cours de l'analyse conceptuelle sont toujours d'actualité. Si des événements survenus dans l'intervalle ont modifié la solution, on retravaille l'analyse conceptuelle, principalement l'activité 1.4 - Déterminer les objectifs et exigences du nouveau système-.

Informations utiles :
- les plans et exigences des utilisateurs (1.4);
- le rapport de la 1e phase et commentaires (1.10);
- les développements politique et économique;
- les développements attendus de l'environnement automatisé;
- les standards et normes.

Activité 2.2: Déterminer le cadre dans lequel le nouveau système devra fonctionner (procédures de travail, environnement software et hardware, cadre organisationnel, conditions d'intégrité, ...) pour prévoir à temps les moyens qui ne sont pas encore disponibles.

Informations utiles :
- les objectifs et exigences du système (1.4);
- la solution sélectionnée (1.8);
- les exigences détaillées et futures (2.1).

Activité 2.3: Diviser le système en sous-systèmes et tâches et les décrire (spécifier les interfaces entre sous-systèmes). La division en sous-systèmes de la phase conceptuelle (activité 1.6 -Faire un schéma du système-) est précisée, éventuellement modifiée, et

définitivement fixée. Les critères pour la séparation en sous-systèmes sont donnés (forte cohérence interne, interfaces réduites).

Informations utiles :
- l'analyse de l'existant (1.3);
- les critères de prestation (1.4);
- les exigences du système (1.4-2.1);
- les fonctions du nouveau système (1.6);
- le cadre du nouveau système (2.2);
- le rapport de la 1e phase (1.11).

Aides possibles :
- la structuration et la statique des traitements;
- SADT (Structured Analysis and Design Technique);
- les diagrammes de flux de données.

Activité 2.4: Définir l'input et l'output par sous-système et par tâche et les interfaces (contenu des écrans et formulaires, dialogues homme/machine, ...) et description des données élémentaires.

Informations utiles :
- les informations sur l'existant (1.2-1.3);
- la description des Entrées/Sorties et des fonctions du nouveau système (1.6);
- les exigences du système, les besoins en information (1.4-2.1);
- le cadre du système (2.2);
- la division en sous-systèmes (2.3).

Aides possibles :
- la statique des traitements;
- le dictionnaire de données.

Activité 2.5/6: Elaborer les diagrammes des traitements (relation entre les tâches) et les descriptions des activités et des tâches (ce qu'elles doivent faire et pas encore le comment). Des critères sont fournis pour aider le choix du mode de réalisation des tâches

(manuelles, automatiques, interactives). Exemples : le nombre de transactions, la fréquence du traitement, le volume des données.

Informations utiles :
- les exigences du système (1.4-2.1);
- la solution sélectionnée (1.8);
- l'environnement du système (2.2);
- la division en sous-systèmes et leur description (2.3);
- la spécification des Entrées/Sorties (2.4).

Aides possibles :
- les diagrammes de flux de données;
- la statique des traitements;
- les tables de décision;
- le prototypage;
- un dictionnaire de données;
- SADT.

Activité 2.7: Spécifier les exigences en sécurité, confidentialité, contrôle, correction, ... (vis-à-vis des manquements du hardware, du software, des personnes).

Informations utiles :
- l'analyse de l'existant (1.3);
- les limites du système (1.5);
- les exigences du système (1.4-2.1);
- le cadre du nouveau système (1.8-2.2);
- la description des sous-systèmes (2.3);
- la description des traitements (2.5/6);
- la législation;
- les techniques de contrôle;
- le dictionnaire de donnée.

Activité 2.8: Déterminer les problèmes humains et des solutions (résistance au changement, problèmes d'ergonomie, dialogue Homme/Machine, utilisation de formulaires, ...). Des critères pour ces aspects organisationnels sont donnés.

Informations utiles :
- les limites et utilitaires (1.5);
- les exigences et objectifs du système (1.4-2.1);
- la description des sous-systèmes (2.3);
- les Entrées/Sorties du système et le dictionnaire de donnée (2.4);
- la description des traitements (2.6);
- les normes ergonomiques, sociales, organisationnelles;
- la législation sur le travail.

Activité 2.9: Concevoir la structure logique des données et les chemins d'accès requis (volume, accès, fréquence, taux de modification, ...).

Informations utiles :
- les Entrées/Sorties du système et le dictionnaire de données (2.4);
- la description des traitements (2.6).

Aides possibles :
- la normalisation;
- un dictionnaire de données;
- la modélisation conceptuelle des données.

Activité 2.10/12: Spécifier les facilités requises en communications de données (temps de réponse, composants du réseau), en hardware (matériel existant, nouveaux composants) et en software (de base, d'application). Cela fera partie du cahier des charges.

Informations utiles :
- les exigences et objectifs du système ainsi que les critères de prestation (1.4-2.1);
- le cadre du nouveau système (1.8-2.2);
- la description des sous-systèmes (2.3);
- les Entrées/Sorties exigées (2.4);
- la description des traitements (2.5/6);
- la sécurité souhaitée (2.7);
- la structure des données, le volume des données, la fréquence d'accès (2.9).

Activité 2.13: Elaborer un plan pour la poursuite du développement, la conversion et la mise en production (conversion, formation, installation, ...); révision éventuelle de l'analyse bénéfices/coûts.

Informations utiles :
- Le rapport de la 1e phase;
- les résultats des différentes activités de la phase 2.

Activité 2.14: Rédiger le rapport d'analyse fonctionnelle. Il sert de base pour prendre la décision de poursuivre, de recommencer une partie ou toute l'analyse ou d'arrêter le développement.

Informations utiles : Tous les résultats obtenus jusqu'ici.

Aide possible : le dictionnaire de données.

2.5. Phase 3 : Analyse technique

2.5.1. Description

L'analyse fonctionnelle a décrit ce que le système doit réaliser, maintenant l'analyse technique décrit comment cela sera réalisé.

Dans les systèmes de moyenne et de grande importance, la distinction entre analyse fonctionnelle et technique est bien marquée car à partir de l'analyse technique, les sous-systèmes peuvent suivre une évolution distincte. Pour les systèmes plus petits, suivant leur complexité, suivant les réalisateurs de l'analyse fonctionnelle, les deux phases peuvent être fusionnées.

Ainsi l'analyse fonctionnelle est détaillée -par sous-système éventuellement- en vue de définir les spécifications relatives à la programmation. C'est ici que l'on définit les procédures de traitement, la structure et l'organisation physique des bases de

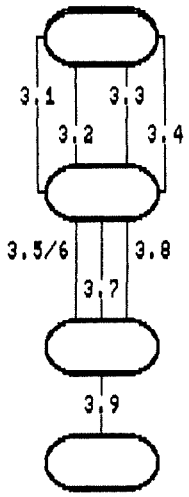
données et des fichiers, les tâches, les critères relatifs au matériel et au logiciel, la découpe en programmes, en modules (Il est important de modulariser au maximum les programmes pour assurer la réutilisabilité de certaines parties et pour faciliter la modification du système.).

L'analyse technique reste aussi indépendante que possible d'outils de réalisation (exemple : du choix d'un langage de programmation, d'une machine).

D'autre part, il est préférable de traiter la partie de l'analyse technique qui concerne l'homme (tâches manuelles, interface) avant la partie machine (programmes), bien que cela puisse en principe se faire simultanément. En effet, il vaut mieux adapter la machine à l'homme et non l'inverse (sinon les utilisateurs risquent de "boycotter" le système par manque de motivation, de degré d'acceptation, ...).

Enfin, à la fin de cette phase, tout doit être spécifié pour que le reste du développement se déroule au mieux. Et, en principe, il ne devrait plus y avoir de retour en arrière pour modifier les spécifications.

2.5.2. Planning possible des activités



Activité 3.1: Conception des procédures manuelles

Activité 3.2: Conception des formulaires et de tous les inputs et outputs de l'ordinateur

Activité 3.3: Conception de la structure de stockage

Activité 3.4: Conception des mesures de sécurité

Activité 3.5/6: Elaboration des descriptions et des schémas de programmes

Activité 3.7: Description des programmes standards à utiliser

Activité 3.8: Elaboration d'un plan détaillé de programmation et de test

Activité 3.9: Rédaction du rapport d'analyse technique

Remarque : la planification et l'exécution de la phase 3 sont souvent combinées avec les phases 4 et 5.

Figure 2.3 : Planning de la phase 3

2.5.3. Description des activités

Activité 3.1: Conception des procédures manuelles, des manuels d'aide ainsi que les moyens d'aide (cours) pour la formation des utilisateurs.

Informations utiles :

- les spécifications des Entrées/Sorties, le dictionnaire de données (2.4);
- la description des traitements, les dialogues (2.5/6);
- les problèmes humains possibles et leurs solutions (2.8);

- les formulaires et les Entrées/Sorties de l'ordinateur (3.2).

Activité 3.2: Conception du design des formulaires et de tous les inputs et outputs de l'ordinateur (design des écrans, listes, imprimés, ...).

Informations utiles :
- les spécifications des Entrées/Sorties, le dictionnaire de données (2.4);
- la description des traitements (2.5/6);
- les aspects ergonomiques (2.8).

Activité 3.3: Conception de la structure de stockage (BD physique, fichiers). Des critères sont donnés pour construire la structure de stockage. Exemples : l'estimation du temps d'exécution, le coût, l'intégrité des données, les problèmes de sécurité. Un spécialiste de la gestion des bases de données est nécessaire.

Informations utiles :
- recueil de données sur l'existant (1.2-1.3);
- les exigences du système (2.1);
- le cadre du système (2.2);
- les spécifications des Entrées/Sorties, le dictionnaire de données (2.4);
- la description des traitements (2.5);
- les exigences en sécurités (2.7);
- la structure des données (2.9).

Activité 3.4: Conception des mesures de sécurité (contrôle, sécurité, confidentialité, solutions de remplacement, reconstitution au démarrage, ...).

Informations utiles : les exigences en sécurité (2.7).

Activité 3.5/6: Elaboration des descriptions et schémas (IPO) des programmes, des modules (partage des écrans, fichiers temporaires, ...). Il faut décrire ce que fait le programme et comment il le fait.

Informations utiles :
- le dictionnaire de données (2.4);
- la description et les schémas des traitements (2.5);
- les procédures (3.1);
- la classification des Entrées/Sorties (3.2);
- la structure de stockage (3.3);
- les mesures de sécurité (3.4).

Aides possibles :
- le pseudo-code;
- les tables de décision;
- la programmation structurée (J.S.P.);
- la statique des traitements.

Activité 3.7: Description des programmes standards à utiliser (routines communes, programmes déjà développés). Exemple : les programmes de tri, de gestion de listes, ...

Informations utiles :
- la description et les schémas des traitements (2.5);
- les spécifications des softwares (2.12);
- les spécifications et les schémas des programmes (3.5).

Activité 3.8: Elaboration d'un plan détaillé de programmation et tests des programmes (données et environnement de test, programmes de test). Exemples de facteurs à prendre en compte : la nature des tâches (batch, interactif), le niveau d'expérience des programmeurs, ...

Informations utiles : Tous les résultats de la phase.

Activité 3.9: Rédaction du rapport d'analyse technique et, éventuellement, révision des bénéfices et coûts.

Informations utiles : Tous les résultats obtenus jusqu'ici.

2.6. Phase 4 : Réalisation des programmes

2.6.1. Description

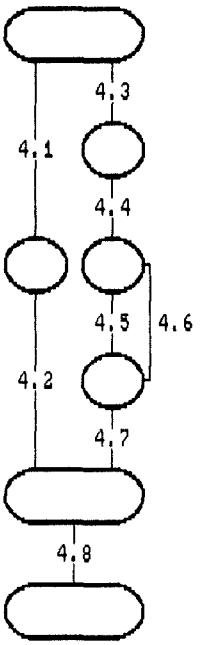
Les programmes sont conçus en vue de l'essai du (sous-)système, sur base des spécifications définies au cours de l'analyse technique. C'est ici que l'on écrit les programmes, essaye ceux-ci et que l'on décrit les tâches manuelles en fonction des procédures manuelles élaborées.

Cette phase consiste à coder, compiler, linker les programmes et à tester leur logique.

En règle générale, un programme peut être éclaté en plusieurs sous-fonctions déterminées de telle manière qu'elles puissent être développées de façon aussi autonome que possible. Il convient donc d'être attentif aux relations entre les différentes parties d'un programme.

Remarque : C'est le dernier moment où l'on peut agir sur la description des tâches et les exigences du système.

2.6.2. Planning possible des activités



activité 4.1: Elaborer les descriptions des tâches

Activité 4.2: Définir les exigences auxquelles le personnel et l'environnement de travail doivent satisfaire

activité 4.3: Détailler les descriptions de programmes

Activité 4.4: Codifier les programmes

Activité 4.5: S'occuper de la traduction et corriger

Activité 4.6: Constituer des données de test

Activité 4.7: Tester les programmes

Activité 4.8: Compléter la documentation de programme

Remarque : Les activités 4.1 et 4.2 sont souvent prévues en phase 6. Les activités 4.3 à 4.7 comprises sont planifiées par programme.

Figure 2.4 : Planning de la phase 4

2.6.3. Description des activités

Activité 4.1: Elaborer les descriptions des tâches manuelles et les normes d'exécution. Des critères sont donnés pour regrouper ou non des procédures de travail en tâches[12] (travail sur les mêmes Entrées/Sorties, facilité d'organisation du travail, ...).

---

12. A ce stade du développement, la notion de "tâche" n'a plus la même signification qu'au niveau fonctionnel (décomposition d'un système en sous-systèmes, activités, tâches). Ici on examine comment l'on va organiser le poste de travail d'un utilisateur.

Informations utiles :
- les facteurs ergonomiques (2.8);
- les descriptions des procédures et des guides-utilisateurs (3.1);
- les formulaires à utiliser et la description des Entrées/Sorties (3.2).

Activité 4.2: Définir les exigences auxquelles le personnel et l'environnement doivent satisfaire (expérience requise, formation, engagement de personnel, exigences techniques, ...).

Informations utiles :
- les facteurs ergonomiques (2.8);
- les descriptions des procédures et des guides-utilisateurs (3.1);
- les formulaires, les utilitaires à utiliser, la description des Entrées/Sorties (3.2);
- la description des tâches (4.1);
- la législation du travail;
- les formations disponibles.

Activité 4.3: Détailler les descriptions de programmes (schémas de la structure des programmes).

Informations utiles :
- la division des écrans, des sorties (3.2);
- la structure de stockage (3.3);
- les spécifications des programmes (3.5/6).

Aides possibles :
- le pseudo-code;
- la statique des traitements pour les modules;
- la programmation structurée (J.S.P.).

Activité 4.4: Codifier les programmes.

Informations utiles :
- les données du dictionnaire (2.4);
- la structure de stockage (3.3);

- les mesures pour le redémarrage (3.4);
- les spécifications des programmes (4.3);
- les manuels de programmation;
- les normes et standards.

Aides possibles :
- les langages de programmation;
- les générateurs de code de programmation;
- les documenteurs de programmes.

Activité 4.5: compiler et corriger (programmes-source, programmes-objet, liste des programmes compilés).

Informations utiles :
- les spécifications des programmes (4.3);
- les programmes codés (4.4);
- les standards de programmation;
- les standards des JCL.

Aides possibles :
- les manuels des différents langages;
- les débuggers.

Activité 4.6: Constituer des données de test des programmes.

Informations utiles :
- la division en enregistrements (3.2);
- la structure de stockage (3.3);
- la structure des programmes (4.3);
- les programmes-source (4.5).

Aides possibles :
- les tables de décision;
- les générateurs de données de test.

Activité 4.7: Tester la logique des programmes et résultats obtenus.

Informations utiles :
- les spécifications des programmes (4.3);

- les programmes-source et les programmes-objet (4.5);
- les données de test des programmes (4.6).

Activité 4.8: Compléter la documentation des programmes (un dossier par programme) et rédaction du rapport final de la phase.

Informations utiles : Toute la documentation sur les programmes.

Aide possible : un dictionnaire de données.

2.7. Phase 5 : Tests

2.7.1. Description

Toutes les parties du système (programmes, procédures, utilitaires utilisés, nouveau matériel) sont testées dans leur interaction.

Les programmes ayant été testés sur le plan de leur logique, ils font à présent l'objet de tests d'intégration dans l'ensemble du système. Il faut, donc, définir des procédures rigoureuses de test. C'est ainsi que l'on peut distinguer différentes sortes de test : de programme, de système, de masse, d'environnement, d'acceptation, car cette distinction fait émerger des erreurs différentes.

La responsabilité du test des procédures de travail incombe aux concepteurs du système chargé de l'installation de celles-ci.

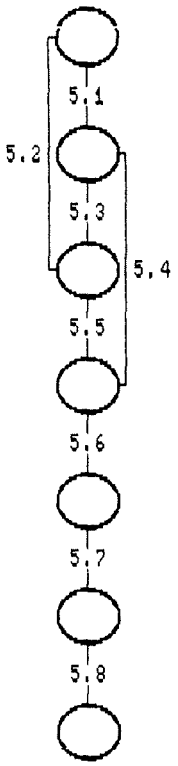
Le test de la cohérence entre les programmes et les procédures est de la compétence des concepteurs mais est souvent réalisé par les programmeurs.

Les informaticiens regardent l'intégration du projet avec les autres applications.

Les demandeurs du projet réalisent le test d'acceptation.

Remarque :
- Un contrôle utilisateur s'effectue lors de toutes les phases à la production du rapport.
- On fusionne parfois les phases 3, 4 et 5 lors de l'utilisation d'outils tel que le prototypage, les langages de 4e génération.

2.7.2. Planning possible des activités



Activité 5.1: Elaboration d'un plan de test détaillé

Activité 5.2: Installation du matériel, du software et préparation de l'environnement de travail

Activité 5.3: Définition des unités de traitement et leur ordre de succession

Activité 5.4: Test des formations, utilitaires et procédures

Activité 5.5: Constitution des données de test de système et d'acceptation

Activité 5.6: Test du (sous-)système

Activité 5.7: Exécution du test d'acceptation

Activité 5.8: Rédaction du rapport "Résultats de test"

Remarque : L'activité 5.2 est en général réalisée beaucoup plus tôt. C'est ici la dernière occasion de réaliser l'installation pour pouvoir effectuer les tests avec le hardware et le software sur lesquels le système sera opérationnel. S'il faut (aussi) installer du matériel dans l'environnement des utilisateurs, ce sera généralement planifié dans le cadre de la phase 6.

Figure 2.5 : Planning de la phase 5

2.7.3. Description des activités

Activité 5.1: Elaboration d'un plan de test détaillé (difficile à estimer car dépendant du nombre d'erreurs

détectées), de procédures de test et de règles concernant la constitution de données de test. On approfondit le plan élaboré au cours de l'activité 3.8 -Elaboration d'un plan de programmation et de test-.

Informations utiles :
- les critères d'acceptation (1.9);
- les critères de prestation et les exigences du système (1.4-2.1);
- les procédures (3.1);
- les mesures et exigences en sécurité (2.7-3.4);
- le plan de test (3.8);
- l'analyse technique (3.9);
- la documentation des programmes (4.6).

Activité 5.2: Installation du matériel, du software et préparation de l'environnement de travail et test (en général, cela est réalisé beaucoup plus tôt).

Informations utiles :
- les critères de prestation (1.4);
- les exigences du système (2.1);
- les facilités requises en communication de données, en hardware et en software (2.10/12);
- le plan d'installation (2.13);
- la littérature spécialisée;
- l'expérience d'autres utilisateurs.

Activité 5.3: Définition des unités de traitement (jobs) et leur ordre de succession (job stream).

Informations utiles :
- les programmes testés (4.7);
- les standards et les normes des JCL.

Activité 5.4: Test des formations (pour la conversion et la mise en production), des utilitaires et des procédures sur un public averti (prérequis pour le test d'acceptation).

Informations utiles :
- les procédures et les manuels-utilisateurs (3.1);
- les formulaires, la description des Entrées/Sorties (3.2);
- la description des tâches (4.1);
- la formation, les utilitaires (4.2).

Activité 5.5: Elaborer des données et des critères de test de système et d'acceptation .

Informations utiles :
- les critères d'acceptation (1.9);
- les critères de prestation et les exigences du système (1.4-2.1);
- la structure de stockage (3.3);
- les mesures de sécurité (3.4);
- les spécifications des programmes (3.6);
- les standards de programmation à utiliser (3.7);
- le plan de test (5.1).

Aides possibles :
- les tables de décision;
- les générateurs de données de test.

Activité 5.6: Test du (sous-)système avec éventuellement les modifications apportées dans l'environnement de travail sous la responsabilité du groupe de projet.

Informations utiles :
- les critères de prestation et les exigences du système (1.4-2.1);
- les mesures et exigences en sécurité (2.7-3.4);
- la description des tâches (4.1);
- les programmes testés (4.7);
- les procédures testées (5.4);
- les données de test (5.5);
- le plan de test (5.7).

Activité 5.7: Exécution du test d'acceptation, avec la liste des défauts constatés, du contenu du système et

non de la conversion et de l'utilisation du système. L'acceptation a lieu après un certain temps d'utilisation et est de la compétence des utilisateurs.

Informations utiles :
- les critères de prestation et les exigences et objectifs du système (1.4-2.1);
- les critères d'acceptation (1.9);
- les mesures et exigences en sécurité (2.7-3.4);
- les données du test d'acceptation (5.5);
- la documentation du système.

Aide possible : les tables de décision.

Activité 5.8: Rédaction du rapport "Résultats de test". Le (sous-)système est accepté ou refusé.

Informations utiles : Tous les résultats des tests.

2.8. Phase 6 : Conversion et mise en production

2.8.1. Description

La mise en production commence dès que le (sous-)système est jugé prêt à l'emploi. La mise en production est pré parée dès la fin de l'analyse fonctionnelle (exemple : acquisition du matériel, programmes de conversion et de lancement, dénomination des fichiers, ...) et se déroule donc en parallèle avec les phase 3 à 5.

La conversion d'un ensemble de données peut s'avérer une partie délicate lors du passage vers le nouveau système. La mise en production peut s'opérer de plusieurs manières :

- Passer d'un seul coup de l'ancien système vers le nouveau,

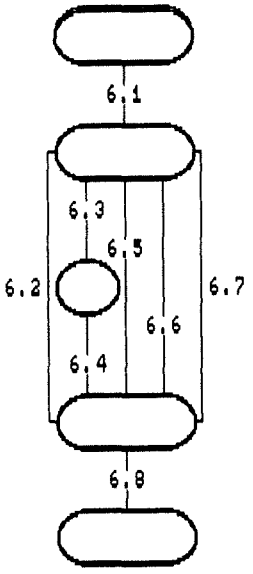
- Lancement progressif,

- Faire tourner l'ancien et le nouveau système en parallèle pendant quelque temps.

Suivant l'ampleur de la conversion, celle-ci peut constituer un projet séparé et être réalisée par un groupe de travail distinct. Elle peut commencer au cours de l'analyse conceptuelle. Bien que son exécution se déroulera plus tard, on peut réfléchir aux problèmes de conversion et au planning de celle-ci au cours de la première phase.

Le développement du système est, à présent, terminé.

2.8.2. Planning possible des activités



Activité 6.1: Elaborer un plan détaillé de conversion et de mise en production
Activité 6.2: Formation du personnel informatique
Activité 6.3: Elaboration d'instructions de conversion et de mise en production
Activité 6.4: Conversion des données
Activité 6.5: Donner des renseignements sur le nouveau système
Activité 6.6: Formation des utilisateurs
Activité 6.7: Formation du groupe de maintenance
Activité 6.8: Mise en production du nouveau système et son transfert

Figure 2.6 : Planning de la phase 6

2.8.3. Description des activités

Activité 6.1: Elaborer un plan détaillé de conversion et de mise en production (plan global de développement éventuellement adapté).

Informations utiles :
- le plan de conversion et de mise en production ainsi que le plan pour l'installation des facilités (2.13);
- les résultats des tests (5.8);
- les possibilités de formation.

Activité 6.2: Formation du personnel informatique (du centre de calcul) et évaluation.

Informations utiles :
- les procédures de reprise (3.4);
- le résultat de l'installation et du test du hardware et du software (5.2);
- les instructions pour les opérateurs (5.3);
- la formation (5.4);
- les normes et standards du centre de calcul.

Activité 6.3: Elaboration d'instructions de conversion et de mise en production (contrôle des données, règles de conversion, matériel nécessaire, ...) et contrôle de la documentation.

Informations utiles :
- les données sur l'existant (1.2);
- les formulaires (2.8);
- les données et la structure de stockage du nouveau système (2.9-3.3);
- les procédures (3.1);
- la description des tâches (4.1);
- les manuels utilisateurs;
- les utilitaires disponibles pour la conversion.

Activité 6.4: Conversion des données et compte rendu des fautes et problèmes constatés.

Informations utiles :
- la structure des données (2.9);
- la structure de stockage, la division en enregistrements, les utilitaires pour la conversion (3.3);
- la description des tâches (4.1);

- le plan de conversion et de mise en production (6.1);
- les instructions pour la conversion (6.3).

Activité 6.5: Donner des renseignements (conséquences sur le personnel, relation avec les autres systèmes, description globale de l'application, ...) sur le nouveau système (réunion et évaluation).

Informations utiles :
- les critères de prestation, les objectifs et exigences du système (1.4);
- les concepts du système (1.8);
- le rapport de la 1e phase (1.11);
- la description des sous-systèmes et des traitements (2.3-2.6);
- le rapport de la 2e phase (2.13);
- les procédures (3.1);
- la description des tâches (4.1);
- le planning pour la conversion et la mise en production (6.1);
- les manuels utilisateurs (6.3).

Activité 6.6: Formation des utilisateurs (donnée par le correspondant informatique si possible) et évaluation.

Informations utiles :
- les procédures (3.1);
- la description des tâches (4.1);
- la formation et les utilitaires (5.4);
- le plan de formation (5.4-6.1);
- les manuels utilisateurs (6.3);
- des cours donnés par l'entreprise ou par les fournisseurs.

Activité 6.7: Formation du groupe de maintenance et évaluation.

Informations utiles :
- le plan de formation (5.4);
- les données de test (5.5);

- la documentation du système (5.7);
- les manuels d'entretien (6.3);
- les standards, les normes, les manuels;
- les cours des fournisseurs, de sociétés de software.

Activité 6.8: Mise en production du nouveau système, son transfert et évaluation. La documentation des (sous-) systèmes est définitive et la documentation du projet est clôturée. Plusieurs approches sont possibles (cfr. infra 2.8.1).

Informations utiles : la documentation du système et du projet.

Aide possible : un dictionnaire de données.

2.9. Phase 7 : Utilisation et gestion du système

2.9.1. Description

Cette phase se prolonge tant que le système est utilisé. C'est ici que l'on verra comment il fonctionne dans la pratique.

Il faut être particulièrement attentif aux procédures de réparation, à la sécurité et aux catastrophes prévisibles car les pannes du système d'information coûtent. Les préparatifs de la sécurité doivent être envisagés dès l'analyse conceptuelle.

Au cours de cette phase, on vérifie périodiquement que le système répond toujours bien à la demande des utilisateurs. Dans la négative, cela peut donner lieu à de nouvelles activités (analyse, modification du dossier d'analyse, modification de programmes, ...).

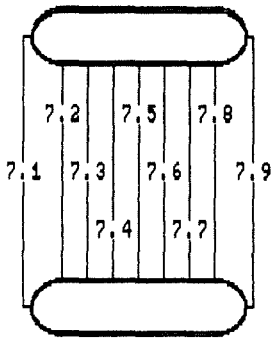
Bien qu'une importante partie des frais entraînés par un système d'informations informatique soient absorbés par cette phase, l'exploitation et la gestion du système sont soumises à très peu de méthodes et de techniques.

On peut s'attendre à ce qu'à l'avenir, lorsque les personnes responsables de la méthode auront acquis l'expérience de très nombreux systèmes, elles pourront généraliser des techniques standards pour mener cette phase à bien.

Les responsables de la gestion du système sont les utilisateurs et le groupe de maintenance pour le système d'information développé et le centre de calcul pour le matériel.

Cette phase termine le cycle de vie du système.

2.9.2. Planning possible des activités



Activité 7.1: Elaboration et utilisation du système de signalement des erreurs

Activité 7.2: Planification périodique de maintenance

Activité 7.3: Planification du traitement informatique

Activité 7.4: Prévention et restauration des fautes et perturbations

Activité 7.5: Surveillance des dispositions de sécurité

Activité 7.6: Modifications et mise à jour de la documentation. Réponse aux besoins en information ad hoc

Activité 7.7: S'occuper des formations complémentaires

Activité 7.8: Gestion des données et des fichiers

Activité 7.9: Evaluation du système et plans d'action

Figure 2.7 : Planning de la phase 7

2.9.3. Description des activités

Activité 7.1: Elaboration et utilisation du système de signalement des erreurs via un canal unique de communication (par exemple, le correspondant informatique).

Informations utiles :
- les exigences du système, les critères de prestation;
- les normes, les standards;
- rapport d'utilisation, nombre de transactions.

Activité 7.2: Planification périodique de maintenance et des activités à réaliser (plan de modification, évaluation du système, personnes nécessaires).

Informations utiles :
- le rapport sur le signalement des erreurs (7.1);
- les propositions de modifications (7.6);
- les formations complémentaires souhaitées (7.7);
- le rapport d'évaluation (7.9);
- la documentation du système.

Activité 7.3: Planification du traitement informatique (planning de production du centre de calcul, des demandes de travaux).

Informations utiles :
- les normes, les estimations du travail actuel et futur;
- les manuels de production.

Activité 7.4: Prévention et restauration des fautes et perturbations (cela implique une bonne documentation, une bonne formation, de bons tests, de bons rapports de perturbation, du personnel disponible, ...).

Informations utiles :
- les rapports sur les problèmes survenus;
- les procédures de reconstruction et de redémarrage;
- les manuels du centre de calcul.

Activité 7.5: Surveillance des dispositions de sécurité et les tester (plan catastrophe, accès au centre de calcul).

Informations utiles :
- les exigences en sécurité (2.7);
- les mesures de sécurité (3.4);
- les rapports sur fautes et les mesures prises (7.4);
- la documentation du système;
- les programmes et techniques de contrôle.

Activité 7.6: Modifications et mise à jour de la documentation. Réponse aux besoins en information ad hoc.

Informations utiles :
- les données de test (5.5);
- le rapport d'évaluation (7.9);
- les souhaits des utilisateurs;
- les possibilités pour corriger;
- de nouveaux appareils.

Activité 7.7: S'occuper des formations complémentaires (objectifs et évaluation). Exemple : une application pratique.

Informations utiles :
- les procédures (3.1);
- la description des tâches (4.1);
- les formations et utilitaires disponibles (5.4);
- les manuels utilisateurs.

Activité 7.8: Gestion des données (base de données) et des fichiers (structure des données et du stockage). Le dictionnaire de données est mis à jour.

Informations utiles :
- le système de signalement (7.1);
- la documentation du système.

Aide possible : un dictionnaire de données.

Activité 7.9: Evaluation du système et plans d'action (rapport périodique, évaluation des coûts d'utilisation et de maintenance, proposition de démarrage d'une nouvelle analyse conceptuelle, ...).

Informations utiles :
- l'analyse bénéfices/coûts (1.10);
- les exigences et objectifs du système (2.1);
- les normes et standards;
- la documentation du système.

2.10. Synthèse

Cette synthèse donne une vue plus globale sur l'évolution des deux composantes principales d'un développement, à savoir les données et les traitements. Nous avons fait de même pour la sécurité car elle est de plus en plus présente dans les entreprises.

2.10.1. Pour les données

Après avoir rassemblé et analysé les données concernées de près ou de loin par le système (activités 1.2 - 1.3), celles-ci sont structurées à l'aide d'un schéma conceptuel des données (activité 1.6). On obtient ainsi une vue globale des données et de leurs relations.

Lorsque la décision de poursuivre le développement est prise, on décrit plus précisément le schéma conceptuel en décrivant les données (activité 2.4), en le normalisant et en le quantifiant (activité 2.9). A partir de ce moment, on peut déterminer les 'clés' d'accès aux données en fonction des traitements (activité 2.9). On obtient ainsi le schéma logique des données.

L'étape suivante, c'est l'implémentation physique des données. On conçoit les entrées/sorties de l'ordinateur et les formulaires (activité 3.2). En même temps, on

construit la structure de stockage des données (activité 3.3).

Ensuite, les programmes sont réalisés et testés, notamment ceux d'accès aux données (activités 4.3 à 4.5). Dans le même temps, on constitue des données de test pour les programmes (activité 4.6).

Lors du stade suivant du développement, on teste le système et, à cette occasion, on sélectionne des données de test (5.5).

Enfin, lorsque le système a été testé et accepté, on procède à sa mise en production. Pour cela, il faut récupérer et convertir les données de l'ancien système (activités 6.3 - 6.4).

Le développement est à présent terminé, mais il faut continuer à gérer les données et leur support (activité 7.8).

2.10.2. Pour les traitements

Après avoir recensé et analysé les données et les traitements concernés de près ou de loin par le système (activités 1.2 - 1.3), les traitements sont structurés et décrits globalement pour mettre en évidence le contenu du nouveau système (activité 1.6). Pour cette réalisation, l'aide de schémas (DFD, SHAT, IPO) est la bien venue.

Lorsque la décision de poursuivre le développement est prise, on décrit les traitements plus précisément en affinant la découpe en sous-systèmes et tâches et en décrivant ceux-ci (activité 2.3). Cela permet de réaliser un diagramme des traitements et de le décrire (activité 2.5/6). On classifie également les différentes tâches détectées (on-line, batch, manuelle, automatique).

Lors de l'étape suivante, on conçoit les procédures de travail autour du système (activité 3.1). Les descriptions des programmes sont élaborées (activité 3.5/6). On met, en particulier, l'accent sur la modularité. De plus, on examine si des programmes standards déjà développés ne sont pas réutilisables (activité 3.7).

Ensuite, les programmes sont réalisés et testés, notamment ceux des traitements (activités 4.3 à 4.5). Dans le même temps, on décrit les postes de travail des utilisateurs (activité 4.1).

Lors du stade suivant, on teste le système et, à cette occasion, on définit des unités de traitement (jobs) et leur ordre de succession (jobs stream) (activité 5.3). On teste également les procédures de travail (activité 5.4).

Enfin, lorsque le système a été testé et accepté, on procède à sa mise en production. Le développement est ainsi terminé, mais l'on continue à contrôler que le système répond toujours bien à l'attente des utilisateurs.

2.10.3. Pour la sécurité

Après avoir rassemblé les données et traitements concernés par le système (activité 1.2), on les analyse (activité 1.3). On regarde notamment quelles données, quelles tâches sont confidentielles et nécessitent donc certaines mesures de sécurité. On classifie les contraintes pour y répondre au mieux (activité 1.4).

Lorsque la décision de poursuivre le développement est prise, on décrit plus précisément les contraintes d'intégrité du système (activité 2.2). A partir de ce moment, on peut spécifier les exigences en sécurité, en confidentialité, ... vis-à-vis des manquements du hardware, du software, des personnes (activité 2.7).

L'étape suivante, c'est la conception des mesures de sécurité (activité 3.4).

Ensuite, les programmes sont réalisés et testés, notamment les programmes de sécurité (activités 4.3 à 4.5).

Lors du stade suivant on teste le système. On contrôle l'inviolabilité des contraintes de sécurité (activités 5.5 à 5.7).

Enfin, lorsque le système a été testé et accepté, on procède à sa mise en production. Le développement ainsi terminé ne clôture pas pour autant le contrôle des contraintes de sécurité (activités 7.4 - 7.5).

Chapitre 3

Les modèles et outils d'aide au développement de logiciels

3.1. Introduction

Une étude critique complète portant sur les modèles et outils d'aide au développement de logiciels constituerait à elle seule un sujet de mémoire volumineux. Aussi n'avons nous pas la prétention de vouloir réaliser ici une telle étude.

Cependant, pour être efficace, une méthode doit reposer sur des modèles et devrait être exploitée à l'aide d'outils logiciels. Si ces deux composantes sont absentes de la démarche, la méthode deviendra fastidieuse voire inutilisable. Dès lors, et vu leur importance croissante, il nous a semblé opportun de réaliser un tour d'horizon des différents modèles et de certains outils.

Les modèles imposés dans le cadre d'une méthodologie de développement sont fréquemment mal ressentis par les développeurs qui voient en eux un contrôle exercé par la hiérarchie.

En outre, ces derniers se justifient souvent en remarquant que, avant l'apparition des modèles, on arrivait quand même à développer des projets comparables en taille à ceux d'aujourd'hui. A leurs yeux, les modèles ne sont donc pas vraiment utiles.

Enfin, les développeurs reprochent souvent aux modèles la perte de temps qu'ils engendreraient. "Les modèles, c'est bien, mais il ne faut pas les utiliser pour le plaisir. Parfois, c'est tellement évident que l'on en n'a pas besoin. Parfois, ils servent. Si on a le temps, on les utilise; si on connaît son affaire, on ne les utilise pas" [propos tenus par un informaticien].

C'est ici qu'apparaît le premier mythe des modèles et des outils. Ils ne sont pas imposés par simple plaisir, dans le seul but de faire perdre du temps à ceux qui les emploient, mais bien d'en gagner à plus long terme.

Ouvrons ici une parenthèse et remettons-nous un instant en tête les données de la crise du logiciel. Alors que les coûts du matériel régressent au fil du temps, ceux des logiciels, eux, ne cessent de croître.

La décroissance des coûts du hardware est due, d'un côté, à l'amortissement, et de l'autre, à la diminution sensible du prix du matériel neuf. En effet, à puissance égale, les prix d'achat décroissent d'à peu près vingt-cinq pour cent par an. Une machine qui coûte aujourd'hui un million sera vendue, dans un an, à plus ou moins sept cent cinquante mille francs. De là, nous pouvons avancer sans trop de risques que, à prix relatif constant, un ordinateur de 1989 sera plus performant qu'un ordinateur de 1988.

Les coûts des logiciels sont, pour leur part, imputables au manque de standardisation au cours du développement, c'est-à-dire à l'absence de méthode adéquate, correspondant au problème traité, et à l'insuffisance (ou à l'inexistence) de documentation.

Ces deux tendances contradictoires risquent fort d'avoir un effet pervers. Considérant les performances sans cesse grandissantes du matériel, les développeurs vont être tentés d'élaborer des logiciels de plus en plus complexes ayant des coûts de maintenance en constante augmentation.

Le but des modèles et des outils n'est donc pas de faire perdre du temps mais bien d'en gagner à plus long terme en diminuant radicalement la maintenance au travers d'une approche de développement plus rigoureuse, plus adéquate, et d'une communication homme - homme optimalisée.

Le second mythe des modèles et outils (et, par extension, des méthodes[13]) est la croyance que, de par leur utilisation, le développement va se dérouler automatiquement, sans nécessiter d'intervention humaine.

Cette conception est totalement erronée. Une technique de modélisation, un outil, ne seront jamais qu'une aide apportée au développeur dans son travail d'analyse. L'élément central restera toujours le raisonnement humain. Si celui-ci est incorrect, aucune technique, aucun outil, ne sera à même de le corriger. Par exemple, un générateur de code recevant en entrée des spécifications incorrectes ne sortira jamais qu'un programme techniquement valable mais fonctionnellement inadéquat.

Ces deux mythes ont heureusement tendance à s'estomper dans l'esprit des développeurs. Afin de la favoriser, il sera toujours nécessaire, lors de l'introduction d'une nouvelle méthode, d'une nouvelle technique, ou d'un nouvel outil, de bien faire comprendre à leurs futurs utilisateurs que leur but est de faciliter le raisonnement, d'améliorer la communication, mais qu'ils ne les remplacent pas.

Dans ce troisième chapitre, nous présenterons une liste (non exhaustive) de modèles et d'outils, pour la plupart utilisés à la C.G.E.R., en essayant, pour chacun d'eux, de cerner leurs domaines d'application privilégiés ainsi que leurs qualités et leurs éventuels défauts. Cette étude se fera sur base de points de vue théoriques, mais aussi en tenant compte des avis de leurs utilisateurs quotidiens : les développeurs.

Nous verrons aussi comment les modèles apportent un support non négligeable à la compréhension et à la

---

13. Nous reviendrons plus longuement sur ce point dans l'ultime chapitre de ce mémoire (Cfr. partie III, chap.2).

communication, et pourquoi les outils constituent un avantage certain.

En fin de chapitre, nous tenterons une classification des modèles et outils dans les différents stades du développement auxquels ils sont destinés.

3.2. Distinction modèles - outils

3.2.1. Les modèles

Un but fondamental des modèles d'aide au développement est la maximisation de la compréhension du réel que l'on tente de modéliser.

En effet, d'innombrables activités peuvent être automatisées, mais la difficulté est toujours de cerner parfaitement la réalité pour que le modèle en soit le plus proche possible.

De ce premier but en découle directement un second : la stimulation du dialogue informaticiens - utilisateurs. N'oublions pas que, dans la plupart des cas, l'utilisateur et l'informaticien sont deux personnes distinctes. Il est, de fait, très rare de trouver un utilisateur (ou un groupe d'utilisateurs) qui développe lui-même sa propre application (application de taille relativement importante bien sûr. Pour les petites applications mono-utilisateur, il leur est toujours loisible, grâce au support d'un infocentre, d'entreprendre eux-même le développement complet).

De ce fait, il est essentiel que l'informaticien comprenne au mieux le problème soumis et explicité par l'utilisateur.

Par rapport au langage naturel qui reste souvent ambigu, les modèles ont l'avantage de formaliser au maximum les éléments du réel.

Comme nous l'avons déjà dit, le but de l'analyse est la construction d'un modèle, restriction du réel, qui ne retiendra de ce réel que les éléments les plus pertinents pour les développeurs. Il est important qu'informaticiens et utilisateurs comprennent d'une manière identique le réel qu'ils étudient ensemble.

Aux niveaux supérieurs de l'analyse, les modèles vont donc formaliser le réel, notamment à l'aide de graphiques, et vont rendre le dialogue informaticiens - utilisateurs le plus limpide possible.

Aux stades plus avancés du développement (analyse technique, programmation), ils vont permettre aux développeurs d'avoir une meilleure approche du problème (par exemple par une analyse "TOP-DOWN") ou encore de travailler avec un langage de plus haut niveau (par exemple en utilisant le langage structuré).

Certains trouveront parfois les modèles lourds à utiliser, notamment dans le cas d'une parfaite connaissance du problème, et préfèrent travailler sans y avoir recours.

Mais aujourd'hui, les logiciels développés en entreprise sont produits par de multiples développeurs et sont destinés à de multiples utilisateurs. La maintenance, quant à elle, est aussi effectuée par des personnes appartenant bien souvent à des groupes différents des deux premiers cités. Il est donc nécessaire d'avoir une approche systématique, au travers d'une méthode et des modèles sous-jacents, afin que le problème et sa solution soient compréhensibles par toute personne touchée de près ou de loin par le projet.

3.2.2. Les outils

Un des arguments favoris des opposants aux modèles est leur lourdeur d'utilisation. Ils n'ont pas tort en ce sens qu'il faut bien souvent prendre crayon et gomme pour réaliser un M.C.D., un D.F.D. ou une structure de type Jackson[14].

Pour évincer cette lourdeur, on a vu progressivement apparaître sur le marché des logiciels facilitant la création de ces différents schémas ou les générant automatiquement. Les plus remarquables d'entre eux sont les générateurs de code qui, à partir de spécifications, très précises il est vrai, produisent un programme exécutable dans un langage défini (COBOL, COBOL II et PL/1 pour le générateur de TELON[15]).

Ces générateurs n'étant pas encore très répandus, il nous semble prématuré de porter un quelconque jugement concernant leur efficacité. Cependant, une remarque concernant les spécifications requises s'impose.

La qualité de bonnes spécifications est la communicabilité. Pour atteindre cette qualité, il est impératif qu'elles soient faciles à comprendre par quiconque se trouvant en contact avec le système. Or, les spécifications nécessaires aux générateurs ne possèdent pas, à notre avis, cette qualité. En effet, elles constitueraient plutôt un langage de (très) haut niveau et ne seraient dès lors plus accessibles à tous.

Pour illustrer cette affirmation, prenons, par analogie, le PROLOG. Ce langage travaille sur base de règles énoncées par les développeurs. Pourtant, malgré cet aspect 'spécification' des règles, il n'en demeure pas

---

14. Cfr. infra 3.3.1., 3.3.5., 3.3.10.

15. Cfr. infra 3.4.3.2.

moins que les développeurs devront bien connaître la manière dont le moteur d'inférence travaille. En cas d'ignorance de ce fonctionnement, les performances des applications produites seront d'une médiocrité inacceptable.

D'après nous, l'utilisation des ces logiciels d'aide au développement va devenir tellement compliquée que seuls des spécialistes pourront les utiliser. L'accessibilité à tous n'est donc pas vérifiée. De plus, tout comme le prix du matériel en baisse constante incite à développer des applications de plus en plus sophistiquées, ces logiciels vont, de par leur apparente simplicité, pousser au développement d'applications de complexité croissante.

De là, le risque d'entrer dans un cercle vicieux n'est pas mince. L'apparition d'outils sophistiqués pousse au développement d'applications plus complexes qui, elles même, nécessiteront de nouveaux outils plus perfectionnés.

Une échappatoire consisterait à former d'abord les gens à réfléchir sans outils, au lieu de leur demander de travailler directement avec eux, afin que, par la suite, ils puissent les utiliser de manière optimale.

Nous verrons dans la suite de ce chapitre que les outils n'apportent pas toujours de solution à tous les problèmes, et que leur incompatibilité et/ou leur complexité d'utilisation est souvent à la base d'une nouvelle résistance de la part des développeurs.

3.3. Les modèles d'aide au développement de logiciels

3.3.1. La modélisation conceptuelle des données

L'approche retenue par le M.A.I. pour construire un modèle conceptuel des données est celle du modèle Entité-Association. Ce choix est motivé par le fait que "Cette modélisation est basée sur un petit nombre de concepts précis (entité, association...) qui offrent une grande souplesse d'adaptation aux cas réels" [MAI 2, p.3].

"Dans une organisation, l'information constitue une ressource vitale au même titre que les personnes, les moyens financiers ou les équipements" [BOPI, II.2]. Dès lors, la qualité de l'information véhiculée revêt une importance toute particulière. L'expansion des systèmes d'informations amène de plus en plus les organisations à gérer la ressource information comme elles gèrent le personnel ou les finances.

La tâche première à réaliser par toute organisation voulant introduire un nouveau système d'information est la définition rigoureuse des éléments qui appartiendront à la mémoire de ce système[16].

Le but d'un modèle tel le modèle Entité-Association est essentiellement de fournir aux développeurs une aide dans leur travail d'identification et de définition des informations qui appartiendront à la mémoire du système.

D'après François BODART et Yves PIGNEUR [BOPI, II.2], les objectifs suivants sont assignés à un modèle de

---

16. La mémoire d'un S.I. est une collection structurée d'informations stockées, décrites par un schéma dit conceptuel. La mémoire représente, à un instant donné, l'état des objets informationnels significatifs pour l'organisation. (inspiré de [BOPI])

structuration des informations (ici le modèle Entité-Association) :

- Aider les concepteurs et analystes à exprimer la sémantique des données contenues dans la mémoire du système, c'est-à-dire à dégager la signification imputée par l'organisation aux données qui représentent des concepts, des objets, des faits ou des événements de l'organisation.

- Aider les réalisateurs du S.I. à comprendre cette sémantique en vue d'organiser un stockage adéquat des données et de créer des procédures d'accès ainsi que des programmes d'application corrects. Dans ce but, un modèle doit posséder une capacité d'expression claire, structurée et lisible des concepts, des objets, des faits et des événements représentés.

- Procurer aux exploitants du S.I. la définition précise des informations qu'ils manipulent ainsi que leurs conditions d'utilisation. A cette fin, un modèle de données devra permettre de dégager non seulement la sémantique attachée à des représentations de concepts, d'objets ou de faits pris isolément mais aussi des représentations d'ensembles structurés d'informations.

- Fournir une représentation opérationnelle des bases de données du S.I. directement exploitables à l'aide de langages d'interrogation de haut niveau.

Pour réaliser ces objectifs, le modèle Entité--Association propose plusieurs concepts de base très simples et facilement utilisables dans la réalité.

L'approche retenue à la C.G.E.R. étant une restriction du modèle proposé par F. BODART et Y. PIGNEUR, nous nous contenterons de reprendre ici les concepts et principes de construction suggérés par le M.A.I. Nous invitons le lecteur intéressé par le modèle complet à lire [BOPI].

3.3.1.1. Définitions

* *ENTITE* (ou type d'entité) : "concept présentant de l'intérêt pour l'entreprise, possédant une existence propre et dont chaque occurrence peut être identifiée de façon univoque par la valeur prise par l'une de ses propriétés appelée IDENTIFIANT" [MAI 2, p.8].

* *ASSOCIATION* (ou type d'association ou relation) : "Concept définissant une relation, une correspondance entre deux[17] ou plusieurs entités" [MAI 2, p.13].
Une association ne peut exister que si les occurrences des entités qu'elle relie existent.

* *ELEMENT* : C'est la plus petite unité d'information manipulée dans l'entreprise. L'élément perdrait toute sa signification s'il était scindé.

* *AGREGAT* : "Ensemble de données considérées comme un tout du point de vue logique. Les agrégats contiennent éventuellement d'autres agrégats, et des éléments" [MAI 2, p.7].

* *IDENTIFIANT* : "Groupe minimal d'attributs et/ou de rôles tel qu'à chaque combinaison de valeurs prises par ce groupe correspond au plus une entité de ce type" [BOPI, II.2.4.3].

* *CARDINALITE* : "La cardinalité d'une association par rapport à une entité sont les nombres minimum et maximum d'occurrences de l'association qui peuvent exister pour une occurrence de cette entité" [MAI 2, p.17].

---

17. Notons que dans le cas d'une association réflexive, les deux entités sont de même type.

Maintenant que les concepts de bases ont été définis, nous allons tenter de décomposer la démarche qui permettra d'aboutir à un modèle conceptuel des données complet et correct. De plus, nous présentons en annexe C un exemple simple d'élaboration d'un tel modèle, ainsi que le graphisme qui le sous-tend.

3.3.1.2. Décomposition de la démarche

a) Elaboration de la liste brute des informations

Cette liste est obtenue à la suite de l'étude de l'existant et des nombreuses discussions qu'ont eues les développeurs et les utilisateurs. Elle doit normalement contenir toutes les informations nécessaires à la modélisation du système.

b) Epuration de la liste brute

La liste établie au cours de l'étape précédente contiendra très probablement des synonymes, des polysèmes et des redondances. L'épuration va fournir une liste des propriétés des objets dans laquelle apparaîtront déjà les éléments et les agrégats.

c) Dégagement des types d'entités

Une même réalité peut être décrite différemment suivant la manière avec laquelle on l'appréhende. Ainsi, un chimiste considérera-t-il une couleur différemment d'un peintre. La construction des entités se fera en fonction des objectifs poursuivis par l'entreprise. Il est important de remarquer que chaque occurrence d'une entité se distingue sans ambiguïté de toute autre occurrence grâce à l'identifiant. Il est donc nécessaire de vérifier

que chacune des entités dégagées possède bien un identifiant au moins.

d) Rattachement de leurs propriétés aux entités

Ce travail a déjà débuté au cours de l'étape précédente via l'adjonction à chaque entité d'un identifiant. Les autres propriétés restantes dans la liste épurée vont être confrontées à la liste des entités pour que chacune de ces propriétés trouve sa place au sein d'une seule entité.

Il se peut que certaines propriétés restent inaffectées. Elles seront alors rattachées ultérieurement aux associations.

e) Définition des relations inter-entités

Dans un S.I., les entités ont des relations (non autonomes) entre elles. Le but de cette étape est de définir les relations qui existent entre les entités.

Tout comme une entité, une association peut être porteuse, c'est-à-dire posséder des propriétés. Il convient donc de rattacher les dernières propriétés de la liste épurée aux associations identifiées.

Remarquons que toute propriété de la liste doit être rattachée exclusivement à une et une seule entité ou à une et une seule association.

f) Déterminer les cardinalités

On va déterminer les cardinalités de chaque couple "entité-relation". Ces cardinalités ont un caractère fondamental en ce sens qu'elles expriment des règles de gestion strictes du S.I.

g) Simplifier éventuellement le modèle

Cette simplification se fera à l'aide de contraintes d'intégrité. Elle doit être réalisée avec la plus grande prudence afin d'éviter toute modification de la sémantique qu'elle pourrait engendrer.

Remarquons quand même qu'un modèle simplifié est nettement plus lisible, et qu'il est plus proche du modèle logique qui sera élaboré par la suite.

h) Quantification du modèle conceptuel

Pour obtenir un modèle statique des données, il suffit de compléter le modèle conceptuel obtenu en adjoignant à chaque objet (entité ou association porteuse) le nombre d'occurrences estimé. Ce modèle est typique à la caisse et il sert de base à l'exécution de PALOMA.

Pour terminer la présentation de la démarche, signalons que la construction d'un modèle conceptuel des données constitue un processus itératif. Chaque fois qu'une nouvelle information, non mentionnée dans la liste brute initiale, apparaît, il est nécessaire de mettre à jour le modèle.

Le nombre d'itérations, et l'importance des modifications qu'elles engendrent, peut varier en fonction de plusieurs critères. Parmi ceux-ci, le degré de connaissance du problème, l'intensité et la qualité des contacts avec l'utilisateur, et l'expérience se révèlent être à nos yeux les plus importants.

3.3.1.3. Domaines d'application du modèle

Un modèle conceptuel des données tel que celui que nous venons de présenter est, par excellence, un support privilégié pour l'analyse conceptuelle des données, mais il peut également être utilisé pour l'étude de l'existant.

3.3.1.4. Avantages d'un modèle E-A

Les avantages d'une telle approche sont nombreux.

Tout d'abord, le modèle conceptuel des données constitue une technique favorisant au maximum le dialogue entre informaticiens et utilisateurs. De par sa simplicité, ces derniers le comprennent facilement et l'acceptent d'autant mieux. De plus, il stimule ce dialogue par ce que nous appellerons une "recherche de la vérité", c'est-à-dire une recherche du modèle correct.

Ensuite, la modélisation résultante est simple et précise. Le système devient compréhensible pour tous et la notion de flou n'existe plus.

Enfin, la souplesse du modèle en fait une technique facilement adaptable à toutes les réalités. Les concepts qu'il met en oeuvre ne sont pas artificiels, ce qui fait que, quelque soit la discipline dans laquelle nous travaillerons, il sera toujours possible de réaliser le schéma.

3.3.1.5. Inconvénients d'un modèle E-A

Un premier inconvénient de ce modèle est que l'appréhension de la réalité n'est pas toujours évidente. Cette difficulté résulte bien souvent d'une mauvaise compréhension du réel, et fait que l'identification des divers objets n'est pas toujours triviale. La décision de créer une nouvelle entité ou une nouvelle relation peut parfois poser problème.

Dès lors, si le modèle n'est pas soigneusement élaboré, il risque d'être corrigé plusieurs fois au cours du développement, pouvant alors entraîner des modifications en cascade sur le reste du système.

D'autre part, alors que la sémantique générale de ce modèle est facilement compréhensible, il n'en n'est pas de même en présence d'associations multiples. En effet, la signification d'une association quaternaire n'est pas toujours évidente pour une personne n'ayant pas participé à l'élaboration du modèle.

Enfin, notons qu'en l'absence d'outil, la réalisation d'un modèle relativement important (comprenant plusieurs dizaines d'objets) est fastidieuse et coûteuse en temps (donc en argent) et que, pour éviter la redondance, l'utilisation de noms parfois peu représentatifs est indispensable.

3.3.2. La normalisation

"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 et la maintenance des bases de données sans influencer la structure logique des données et les programmes d'application" [MAI 5, p.1].

Bien que les modèles issus de l'analyse conceptuelle soient quelquefois en forme normale avancée ("bon sens inspiré"), il est utile de vérifier le degré de normalisation et, si nécessaire, de le compléter.

Notons que la normalisation est quasi indispensable lorsque l'on utilise un S.G.B.D. de type relationnel.

Le processus de normalisation peut être poussé plus ou moins loin. Jean-Luc HAINAUT [HAI 1, p.5.6] dénombrait six formes normales différentes. Dans cet exposé, nous ne retiendrons que les formes normales reprises dans les cours du M.A.I. Le lecteur intéressé pourra se référer à [CODD] et à [HAI 1].

Remarquons que la normalisation constitue un sujet d'étude très vaste qu'il n'est pas possible de reprendre intégralement ici. Nous nous efforcerons pourtant d'en présenter les aspects les plus importants tout en restant malheureusement incomplet.

3.3.2.1. Concepts de base

La notion d'*identifiant* est rigoureusement la même que celle donnée pour le schéma Entité-Association. Notons cependant qu'une relation (voir ci-dessous) peut contenir plusieurs identifiants non--nécessairement disjoints, mais qu'elle en contient toujours au moins un. Par exemple, l'ensemble des attributs de toute relation est un identifiant.

Une remarque s'impose au sujet des identifiants. En fait, le passage d'un schéma E-A vers une modèle relationnel n'est possible que grâce à la notion d'identifiant strict introduite. En effet, sans celle-ci, il se pourrait fort bien qu'au niveau conceptuel l'on aie deux occurences identiques d'une même entité qui, une fois transposées au modèle

relationnel, ne constitueraient plus qu'un et un seul tuple.

Un *identifiant strict* est un (groupe d') attribut(s) qui ne contient(-nnent) pas de sous-ensemble d'attributs qui soit encore identifiant.

Les *dépendances fonctionnelles* (DF) jouent un rôle important dans le modèle relationnel. "Une DF d'un attribut A vers un attribut B dans une relation R exprime que chaque fois qu'une valeur de A apparaît dans R, il lui est associé la même valeur de B. Une DF peut aussi exister d'un groupe d'attributs vers un autre" [HAI 1, p. 1.4]. B dépend alors (fonctionnellement) de A et B est déterminé par A, ou encore A détermine B.

Une *dépendance fonctionnelle stricte* "est une DF qui cesse d'en être une si on lui enlève l'un quelconque de ses attributs déterminants" [HAI 1, p. 4.7]. Pour l'exemple ci dessus, on dirait alors que A détermine strictement B, que B dépend strictement de A, ou que B est déterminé strictement par A.

Ainsi, tout identifiant est source de DF vers les autres attributs de la relation.

A côté des DF, il existe les *dépendances multivaluées* (DM). "Il existe dans une relation R une DM de l'attribut A vers l'attribut B si quelles que soient les valeurs des autres attributs de R, à une valeur déterminée de A est toujours associé le même sous-ensemble de B. En particulier, un DF est une DM" [HAI 1, p. 1.4].

Les DF ont, entre autres, la propriété de transitivité. Ainsi, si dans une relation R un attribut A détermine un attribut B, et que B détermine lui-même un attribut C, alors A détermine C.

a) La représentation des informations

Intuitivement, une relation (à ne pas confondre avec un association dans un modèle E-A) est une table dans laquelle chaque ligne correspond à une occurrence d'un objet, c'est-à-dire une entité ou une association, du modèle conceptuel des données développé en début d'analyse. Le nom de la table est celui de l'entité, et chaque colonne correspond à un attribut de l'entité. Ces attributs sont définis sur des domaines de valeurs possibles. La clé primaire de la relation est constituée de l'identifiant principal de l'entité.

Si le modèle conceptuel a été mal construit, on aboutira à des tables problématiques où apparaîtront des anomalies. De ce fait, moins le modèle conceptuel sera soigné et plus le processus de normalisation sera important, et inversement.

A la C.G.E.R., la normalisation se compose de plusieurs formes normales successives dont les trois premières sont fortement conseillées[18].

b) La première forme normale (1FN)

Une relation est sous 1FN si elle est définie sur des domaines simples (s'il n'existe aucun attribut répétitif).

Malgré cette 1FN, des anomalies de redondance peuvent subsister. Les causes de celles-ci sont :

- l'identifiant n'est pas un déterminant strict d'un attribut non identifiant;

---

18. Un exemple complet de normalisation est présenté en annexe D.

- un attribut non identifiant dépend d'une partie seulement de l'identifiant.

Pour éliminer cette redondance, on a recours à la deuxième forme normale.

c) La deuxième forme normale (2FN)

Une relation est sous 2FN si elle est sous 1FN et si tout attribut non identifiant dépend strictement de chacun des identifiants stricts.

Encore une fois certaines redondances ne sont pas éliminées. Leurs causes sont :

- un attribut non identifiant dépend d'attributs non identifiants;

- un attribut extérieur à l'identifiant dépend transitivement de l'identifiant.

d) La troisième forme normale (3FN)

Une relation est sous 3FN si elle est sous 2FN et s'il n'existe aucune dépendance transitive entre un identifiant et un attribut non identifiant via des attributs non identifiants.

Comme nous l'avons déjà remarqué plus haut, la 3FN représente une sorte de "bon sens inspiré", et la normalisation se résumera parfois à une simple vérification que le modèle conceptuel des données est bien en 3FN.

Dans certains cas cependant, le processus peut être poussé plus loin. Nous avons alors une quatrième et une cinquième forme normale.

e) La quatrième forme normale (4FN)

Une relation est sous 4FN si elle est sous 3FN et si elle ne contient pas plus d'un attribut indépendant à valeurs multiples.

f) La cinquième forme normale (5FN)

Outre la 4FN, le M.A.I. propose la 5FN. Nous n'avons trouvé aucune définition précise de cette 5FN dans la documentation volumineuse issue du service. Néanmoins, d'après l'exemple fourni dans le cours de la C.G.E.R., elle nous semble très proche d'une forme normale présentée par Jean-Luc HAINAUT [HAI 1, p.5.19] et qui en donne la définition suivante : "Une relation sous 1FN est sous [5FN] si toute dépendance multivaluée non triviale est une dépendance fonctionnelle".

3.3.2.2. Domaine d'application de la normalisation

Typiquement, le processus de normalisation sera appliqué au modèle conceptuel des données après l'élaboration de celui-ci. Déterminer si la normalisation doit se dérouler en phase conceptuelle ou en phase fonctionnelle relève, pour nous, du détail. Ce qu'il importe de signaler est que cette normalisation va permettre d'établir le schéma logique de la base de données et doit donc être effectuée auparavant.

3.3.2.3. Avantages de la normalisation

L'avantage de la normalisation est que, si elle est poussée suffisamment loin, elle élimine pratiquement toute redondance. De ce fait, peu de problèmes risquent de survenir lors de la mise-à-jour, de la suppression ou de l'insertion d'éléments.

De plus, si le modèle conceptuel des données a été soigné, le processus de normalisation peut être ramené à une simple vérification.

3.3.2.4. Inconvénients de la normalisation

La volonté d'éliminer à tout prix la redondance peut conduire à un système de relations plus complexe. Du fait de cette complexité, le schéma logique qui en résultera risquera de ne pas répondre à tous les critères d'efficacité du S.G.B.D. employé. Il faudra donc souvent trouver le juste milieu entre d'une part une volonté d'éliminer toute redondance et, d'autre part, un soucis d'efficacité maximum.

D'autre part, la normalisation se base uniquement sur les données et ne tient jamais compte des traitements effectués sur celles-ci. Or, il se peut qu'un base de données non normalisée soit bien plus performante qu'une autre base de données travaillant sur les mêmes données, normalisée, et nettement moins volumineuse. De plus, l'éclatement des relations peut, lors de l'utilisation du système, nécessiter, d'une part, l'ouverture d'un bon nombre de fichiers différents (si le système d'exploitation limite le nombre d'ouvertures simultanées, des problèmes risques d'apparaître) et, d'autre part, des comparaisons inter-fichiers importantes et donc coûteuses[19].

---

19. Cfr. exemples en annexe D.

La clairvoyance du concepteur devra donc lui permettre de développer une base la plus performante possible en évitant éventuellement des redondances inutiles.

3.3.3. Le modèle dynamique des données

A la C.G.E.R., le modèle dynamique des données résulte directement de l'exécution de PALOMA[20]. Il visualise les points d'entrée dans le modèle conceptuel et estime le trafic inter-objets. De par ces estimations, il aide le D.B.A. dans sa tâche de transformation du modèle conceptuel en un modèle logique des données.

Les résultats représentés par ce modèle dynamique sont très critiquables. En effet, les différentes données utilisées par PALOMA ne sont que des estimations grossières.

Par soucis de clarté, nous ne nous attarderons pas d'avantage sur ce modèle ici. Nous en continuerons la critique lorsque nous analyserons en détail PALOMA.

3.3.4. Le modèle logique des données

D'après nos conversations avec le D.B.A., il est ressorti qu'il n'existait à la Caisse aucune technique de transformation du schéma conceptuel des données en un schéma logique. De ce fait, seuls l'expérience et l'instinct permettent au D.B.A. de parvenir à un schéma logique correct et optimal. A sa demande, nous lui avons procuré le livre de Jean-Luc HAINAUT [HAI 2] traitant de ce problème.

Dans l'exposé qui suit, nous présentons succinctement le modèle d'accès généralisé (M.A.G.) développé dans l'ouvrage précité.

---

20. Cfr. infra 3.4.2.

3.3.4.1. Les objets de base du modèle

* *ARTICLE* : "L'article est une entité d'information, stockée dans la base de données, qui peut être créée ou supprimée, et à laquelle il est possible d'accéder" [HAI 2, p.21]. Tous les articles de la base de données sont distincts et appartiennent à un et un seul *TYPE D'ARTICLES* qui en définit les propriétés communes.

A chaque article peuvent être associés 0,1 ou plusieurs domaines (type de données) jouant le rôle d'*ITEM*. Chaque item porte un nom qui l'identifie parmi les items d'un type d'articles.

* *ITEM ELEMENTAIRE* : Item défini sur un domaine de valeurs atomiques, indécomposables. La fragmentation de ces valeurs leur ferait perdre toute signification dans le contexte du système d'information.

* *ITEM DECOMPOSABLE* : Item dont la valeur est une suite de valeurs significatives appartenant chacune à un domaine composant de l'item décomposable.

* *ITEM SIMPLE* : Item dont une seule valeur peut être affectée à chaque article.

* *ITEM REPETITIF* : Item dont plusieurs valeurs peuvent être affectées à un même article.

* *ITEM OBLIGATOIRE* : Par opposition aux *ITEMS FACULTATIFS*, "un item d'un type d'articles est obligatoire si à tout article est associé au moins une valeur de cet item" [HAI 2, p.23].

* *ITEM IDENTIFIANT* : Un item est identifiant si à toute valeur de cet item n'est associé qu'au plus un article.

* *CHEMIN D'ACCES* : "Le chemin d'accès inter-articles est un mécanisme 1) qui associe à un article (appelé origine du chemin) une suite de 0,1 ou plusieurs articles (appelés cibles du chemin), 2) tel qu'il est possible, à partir de l'article origine, d'accéder aux articles cibles successifs du chemin" [HAI 2, p.23]. Tout chemin d'accès appartient à un seul *TYPE DE CHEMINS* qui en définit les propriétés communes.

* *CLASSE FONCTIONNELLE* : Une classe fonctionnelle décrit le nombre maximum d'articles origine et le nombre maximum d'articles cible des type de chemins.
-> *1-N* : un chemin peut contenir de 0 à N cibles, mais chaque cible ne peut l'être que d'un seul chemin.
-> *N-1* : un chemin ne peut contenir qu'une seule cible, mais une cible peut l'être de plusieurs chemins.
-> *1-1* : un chemin ne peut contenir qu'une seule cible, et chaque cible ne peut l'être que d'un seul chemin.
-> *N-N* : un chemin peut contenir de 0 à N cibles et une cible peut l'être de 0 à N chemins.

* *CLE D'ACCES* : "Une clé d'accès est un item, ou un groupe d'items d'un même type d'articles, tel qu'il existe un mécanisme permettant d'accéder successivement aux articles auxquels est associée une valeur déterminée de cette clé, et à eux seulement" [HAI 2, p.27].

Il existe un langage de désignation de données permettant de désigner des sous-ensembles de données (items et articles). Ce langage s'appuie sur la description des données faite par le M.A.G.. Il permettra notamment de réaliser les futures requêtes base de données à incorporer aux programmes.

Un M.A.G. pourra subir plusieurs modifications afin de le rendre conforme à un type de S.G.B.D. particulier. Les transformations nécessaires n'engendreront bien souvent que peu de modification.

3.3.4.2. Domaine d'application du M.A.G.

Le M.A.G. "décrit les structures de données non seulement sous l'angle de la sémantique qu'expriment ces données, mais aussi sous celui des accès dont elles peuvent faire l'objet. En ce sens, ce modèle ne peut être qualifié de conceptuel" [HAI 2, p.20].

Le M.A.G. sera mis au point sur base du modèle conceptuel des données et éventuellement de la normalisation. Si des calculs d'estimation de trafic et d'accès aux données plus ou moins précis sont effectués (PALOMA à la C.G.E.R.), ils seront d'un grand secours pour l'élaboration d'un M.A.G. performant.

Comme le précise Jean-Luc HAINAUT, ce modèle faisant appel à des notions d'accès et d'appel ne peut être considéré comme conceptuel. De plus, le M.A.G. n'est pas encore physique. Dès lors, son domaine d'application est plutôt l'analyse fonctionnelle.

3.3.4.3. Evaluation du M.A.G.

Nos contacts avec le M.A.G. ne sont que théoriques. Les quelques exercices réalisés lors de travaux pratiques ne sont pas suffisants à nos yeux pour nous permettre de critiquer ce modèle. Pourtant, plusieurs considérations peuvent être avancées.

Tout d'abord, les concepts mis en oeuvre par ce modèle et le modèle conceptuel des données sont relativement proches. Cette caractéristique va faciliter

grandement la construction d'un premier schéma M.A.G. qui ne sera plus qu'une traduction évoluée du modèle conceptuel.

Ensuite, le M.A.G. est multi-cibles. Cette caractéristique autorise un usage très vaste de ce modèle. Si un S.G.B.D. est déjà connu, le M.A.G. pourra encourir quelques modifications qui le rendront conforme à ce S.G.B.D.

Enfin, quoique cette démarche ne soit pas conseillée, si le système à développer est de petite taille, et si le développeur connaît bien le réel abordé, un M.A.G. peut être réalisé directement d'après ce réel, sans passer par un schéma E-A. Si le développeur est sûr de lui, un temps appréciable peut être gagné. Mais insistons sur le fait que cette démarche est très dangereuse et ne devra être adoptée que dans des cas exceptionnels.

3.3.5. Le Diagramme de Flux de Données

Un diagramme de flux de données donne une représentation graphique du fonctionnement de l'entreprise (ou d'une partie de celle-ci), et plus particulièrement des flux d'information circulant en son sein.

Le modèle que nous reprenons ci-dessous est celui qui est prôné par le M.A.I.

3.3.5.1. Les éléments du diagramme

Pour représenter un D.F.D., quatre objets de base sont proposés[21] :

---

21. Cfr. annexe E.

- Les *FLUX* représentent l'information circulant dans l'entreprise. Un flux peut être un coup de téléphone, un document ou encore un support magnétique. L'information prise en compte n'est pas le document en lui-même ou le support magnétique, mais bien les informations reprises sur ces supports et pertinentes pour l'organisation. Par exemple, pour un bon de commande, les informations retenues seront : le numéro du bon, la date, le numéro du client et les numéros et quantités des produits commandés.

- Les *TRAITEMENTS* décrivent des transformations d'un ou plusieurs flux.

- Les *STOCKAGES* sont les informations qui sont momentanément stockées et qui risquent de subir un traitement dans le futur.

- Les *ACTEURS EXTERNES* sont les sources ou destinations externes des flux du systèmes.

La technique de construction utilisée est une approche par raffinements successifs.

Tout d'abord, le système sous-étude est considéré dans sa globalité. On a alors un "diagramme de contexte". Notons au passage que ce diagramme de niveau supérieur est très important car il permet de mieux apprécier les limites du système.

Ensuite, itérativement, le diagramme de contexte va être partitionné en niveaux d'activités de plus en plus simples. A la fin de la décomposition, les traitements résultants seront élémentaires et assimilés aux tâches.

La décomposition effectuée ici, quoique similaire à celle effectuée par une S.H.A.T.[22] correspond à une

---

22. Cfr. infra 3.3.6.

découpe horizontale du système, alors qu'une S.H.A.T. est une découpe verticale. Nous reviendrons plus en détail sur ces deux découpes lors de la présentation de cet autre modèle.

Dans un D.F.D., aucune information n'est créée ni perdue. Elle se transforme (insistons sur le fait que l'information n'est ni le document, ni le message, ni le coup de téléphone, ni le support magnétique. Elle est constituée des éléments pertinents contenus dans ces divers médias. La transformation consiste, d'une part, en une déduction d'informations nouvelles à partir des informations initiales, et, d'autre part, en un changement de présentation de celle-ci en utilisant les supports les plus adéquats.). De ce fait, tout élément qui sort du système doit y avoir été introduit ou doit résulter d'une déduction au sein même de celui-ci. De plus, tout élément introduit dans le système ne peut être inutile.

Ces deux dernières propriétés sont utilisées pour vérifier la cohérence et la complétude du système élaboré.

3.3.5.2. Domaines d'application d'un D.F.D.

Un D.F.D. est particulièrement adéquat pour l'analyse conceptuelle des traitements (y compris l'étude de l'existant). Il donne une vision tantôt globale, tantôt détaillée du système. De plus, il peut constituer une aide non négligeable dans la définition des frontières du système.

3.3.5.3. Avantages d'un D.F.D.

Un D.F.D décrit au travers d'un seul modèle plusieurs aspects d'un système.

Il décrit les circuits d'information existants dans le système et dans l'entreprise de façon simple mais efficace.

De plus, il donne une représentation de la succession des tâches à effectuer pour accomplir un traitement complet, exprimant ainsi la dynamique du système.

Enfin, il décrit succinctement la statique du système en faisant ressortir les éléments nécessaires à l'exécution d'une tâche.

3.3.5.4. Inconvénients d'un D.F.D.

Vu sa grande richesse, un D.F.D. peut devenir vite lourd et poser des problèmes de compréhension. Sa réalisation risque d'être une grande consommatrice de temps.

Un D.F.D. n'est pas non plus une représentation complète du fonctionnement de l'entreprise. Il décrit bien l'enchaînement des traitements et les informations circulant aux travers de ceux-ci, mais il omet de définir ces informations. Par exemple, si le flux est un bon de commande, il est utile de spécifier quels sont les éléments repris sur le bon qui sont nécessaires aux différents traitements.

Pour être pleinement efficace, un D.F.D. devrait donc être complété par une description précise de tous les éléments qui transitent au travers.

Pour terminer, signalons que la présentation graphique retenue à la C.G.E.R. peut devenir assez floue lorsqu'un système de grande dimension est représenté. Néanmoins, des variantes de D.F.D. proposant une représentation plus lisible existent, notamment chez [BOPI].

3.3.6. La Structure Hiérarchique des Activités et Tâches

3.3.6.1. Présentation de la S.H.A.T.

Cette technique constitue une approche "top-down" du problème que l'on décrit au moyen d'une découpe verticale. Le problème initial est décomposé en sous-problèmes plus simples qui sont à leur tour décomposés en sous-problèmes plus élémentaires, et ainsi de suite.

Le résultat est une arborescence où tout élément intermédiaire de niveau i (i>1) provient de la décomposition d'un élément de niveau i-1 et est décomposé en n (n>=1) éléments de niveau i+1.

Dans cette arborescence, les éléments sont caractérisés par leur niveau de détail. Ainsi, on identifiera le système, les sous-systèmes, les activités et les tâches.

* *SYSTEME* : Un système est "un ensemble de données et de traitements possédant une grande cohérence interne, et un minimum d'interfaces avec l'extérieur" [MAI 3, p.5].

* *SOUS-SYSTEME* : Un sous-système est une "unité susceptible d'être développée et mise en production isolément" [DOL, module 1]. Les critères d'identification d'un sous-système sont la cohésion interne (la plus forte possible), l'interfaçage avec les autres sous-systèmes (le plus faible possible) et la fonctionnalité du sous-système (il traite d'un problème précis).

* *ACTIVITE* : L'activité est "un ensemble cohérent de tâches nécessaires à l'accomplissement d'une finalité permanente et essentielle du système" [DEDE, p.12].

* *TACHE* : "La tâche constitue une unité de traitement aux yeux de l'utilisateur, un ensemble organisé d'opérations élémentaires transformant selon des règles pré-définies des informations d'entrée en informations de sortie" [MAI 3, p.13].

Le schéma repris en annexe F montre les relations existant entre ces différents éléments.

Remarquons que l'arborescence contiendra généralement plus de quatre niveaux. Il y aura bien souvent plusieurs niveaux de type "activité" (d'où la définition assez large). Par contre, les éléments terminaux ne correspondront jamais qu'à des tâches, et le racine de l'arborescence aura toujours les caractéristiques d'un système.

Comme nous l'avons signalé lors de la présentation du D.F.D., la S.H.A.T. représente une découpe verticale du système alors que le D.F.D. correspond plutôt à une approche horizontale. Pourtant, les deux découpes sont complémentaires. A un niveau de la S.H.A.T. se rapporte un raffinement du D.F.D. Par exemple, le diagramme de contexte correspond au système, et l'ultime raffinement du D.F.D. se rapporte aux différentes tâches du dernier niveau de la S.H.A.T.

3.3.6.2. Domaine d'application d'une S.H.A.T.

La découpe du système doit s'effectuer au niveau de l'analyse conceptuelle. La S.H.A.T. sera utilisée à ce niveau pour identifier les différents sous-systèmes, et, dans chaque sous-système, pour arriver aux tâches.

3.3.6.3. Avantages d'une S.H.A.T.

Une S.H.A.T. est particulièrement efficace pour décomposer le système en sous-systèmes ou pour analyser un sous-système de taille raisonnable. La compréhension du système est quasi immédiate grâce au formalisme graphique simple.

3.3.6.4. Inconvénients d'une S.H.A.T.

Une telle décomposition devient vite lourde quand il s'agit d'un gros système. En effet, les niveaux risquent de se multiplier, et le nombre de tâches terminales augmente de façon impressionnante. Dès lors, il n'est plus possible de l'utiliser efficacement comme moyen de visualisation globale de tout le système.

De plus, de par sa dimension et le manque d'automatisation pour la réaliser, l'arborescence sera souvent délaissée par les développeurs.

3.3.7. Le diagramme Input Process Output.

Une fois la découpe du nouveau système effectuée (via un D.F.D. et une S.H.A.T.), la spécification de chacune des tâches peut commencer.

Celle-ci s'effectue à la Caisse d'Epargne via un diagramme I.P.O. (Input - Process - Output). La documentation le concernant (à la C.G.E.R.) est très maigre, aussi nous efforcerons-nous d'être le plus précis possible malgré ce manque d'informations.

Le diagramme I.P.O. ressemble à une simplification du modèle de la statique des traitements proposé par F. BODART et Y. PIGNEUR [BOPI]. Son état actuel d'avancement à la C.G.E.R. pourrait, à notre avis, être

progressivement amélioré (ou mieux défini) pour parvenir à un modèle proche de celui proposé par les deux auteurs pré-cités.

3.3.7.1. But d'un I.P.O.

Le but d'un diagramme I.P.O. est de décrire le plus complètement possible des traitements d'un même niveau, à savoir les tâches, en identifiant les informations qui leur sont nécessaires en entrée et les informations produites en sortie. De plus, un I.P.O. décrira plus ou moins précisément le traitement qui permet la transformation des entrées en sorties.

Le diagramme sera complété plus tard (en analyse fonctionnelle) par l'adjonction de l'origine des informations en entrée et de la destination des informations en sortie.

Précisons qu'un traitement ne peut recevoir en entrée que des informations issues du modèle conceptuel des données, c'est-à-dire stockées dans la mémoire du S.I. Il en va de même pour les informations en sortie. Cependant, il se peut qu'un document porteur d'informations soit aussi nécessaire à l'exécution d'une tâche, ou qu'une information non stockée circule de tâche en tâche. Ces éléments seront aussi repris dans un I.P.O.

3.3.7.2. Présentation d'un I.P.O.

Le diagramme I.P.O. se présente sous la forme d'un tableau à trois ou cinq colonnes[23].

---

23. La représentation d'un I.P.O. et quelques exemples se trouvent en annexe G.

- La première donne, pour chaque élément en entrée, son origine.

- La deuxième colonne décrit les inputs, c'est-à-dire les éléments nécessaires à l'exécution de la tâche. Dans le cas d'éléments appartenant au modèle conceptuel des données, on donnera leur nom.

- La troisième colonne décrit le traitement effectué pour obtenir les sorties à partir des entrées. Cette définition des outputs en fonction des inputs se fera soit en pseudo-code[24], soit d'après la technique J.S.P.[25], soit en utilisant les tables de décision[26]. Elle sera plus ou moins poussée suivant le niveau d'analyse abordé. Comme un I.P.O. est un modèle de spécification, il est important que les règles exprimant le traitement soient libres de toute contrainte.

- La quatrième colonne contiendra les éléments fournis en output par la tâche.

- La dernière donne, pour chaque élément en sortie, sa destination.

Remarquons que les première et dernière colonnes sont surtout utilisées en analyse fonctionnelle.

L'origine et la destination d'un élément peuvent être soit la mémoire du système, soit une autre tâche, soit un acteur externe (dans le cas d'un document ou d'informations non mémorisées).

---

24. Cfr. infra 3.3.8.

25. Cfr. infra 3.3.10.

26. Cfr. infra 3.3.9.

3.3.7.3. Domaines d'application d'un I.P.O.

Un tel diagramme peut être utilisé au cours de l'étude de l'existant et aux niveaux de l'analyse conceptuelle et de l'analyse fonctionnelle.

Lors de l'étude de l'existant, un I.P.O. "simple", c'est-à-dire à trois colonnes, et dont la seconde défini sommairement le traitement, permettra de mieux comprendre le fonctionnement du système et de chaque tâche en particulier.

Au cours de l'analyse conceptuelle, le I.P.O. constituera un complément idéal des D.F.D. et S.H.A.T. pour esquisser les système.

Enfin, en analyse fonctionnelle, un I.P.O. complet, à cinq colonnes, donnera une spécification précise des tâches. La spécification du traitement à effectuer pour obtenir les outputs à partir des inputs sera précise, mais non procédurale.

3.3.7.4. Avantages d'un I.P.O.

Un diagramme I.P.O. a l'avantage d'individualiser chaque tâche en spécifiant précisément tous les éléments nécessaires à son exécution. Il offre aussi la possibilité de spécifier le traitement en utilisant des techniques telles les tables de décision.

3.3.7.5. Inconvénients d'un I.P.O.

Un I.P.O. simple est redondant par rapport à un D.F.D. Dès lors cette redondance constituera un excellent prétexte pour les développeurs afin de ne pas l'utiliser.

La technique de spécification du traitement est très importante. Le pseudo-code figure parmi les possibilités. Cependant, celui-ci induit déjà l'idée d'algorithme, alors qu'un I.P.O. n'est utilisée que jusqu'en analyse fonctionnelle. Dès lors, par cette technique, la spécification du traitement devient solution. Or, seules des spécifications non algorithmiques sont acceptables. Il convient donc d'éviter le pseudo-code pour spécifier les traitements d'un I.P.O.

3.3.8. Le pseudo-code ou langage structuré

Le pseudo-code constitue une sorte de dégénérescence du langage courant, un style télégraphique. Il est caractérisé par l'emploi d'une syntaxe très réduite et d'un vocabulaire précis lui conférant une orientation fortement algorithmique. De ce fait, il est plus une technique de description de la solution (à savoir les procédures et programmes) plutôt qu'une technique de spécification fonctionnelle des traitements.

3.3.8.1. La syntaxe du langage

Le langage utilise les trois structures de base que sont la séquence, l'itération et la sélection. Il propose également à ses adeptes des actions telles que "lire" et "écrire" qui autorisent l'idée d'accès à des fichiers.

Pour la séquence, on utilise le terme DO :

```
DO
...(séquence)...
ENDDO
```

Pour l'itération, le terme LOOP est employé :

```
LOOP
WHILE/UNTIL (condition)
...(séquence)...
ENDLOOP
```

Enfin, pour la sélection, on a soit le IF traditionnel ou une structure de type CASE en PASCAL :

```
IF (condition)
(THEN)
...(séquence)...
ELSE
...(séquence)...
ENDIF
```

ou :

```
SELECT
WHEN (condition 1)
...(séquence)...
WHEN (condition 2)
...(séquence)...
WHEN (condition 3)
...(séquence)...
ENDSELECT
```

Des exemples simplifiés de description d'algorithmes en pseudo-code sont présentés en annexe H.

3.3.8.2. Domaine d'application du pseudo-code

Le domaine d'application privilégié du pseudo-code est l'analyse technique des traitements. D'après les diagrammes I.P.O. spécifiés en analyse fonctionnelle, il va permettre de décrire parfaitement et précisément les traitements qu'il ne restera alors plus qu'à traduire dans le langage de programmation cible.

Un facteur restrictif serait la présence d'un nombre important de conditions dans un algorithme. Dans ce cas, l'emploi du langage structuré est déconseillé, car source d'erreurs.

3.3.8.3. Avantages du pseudo-code

L'avantage principal du pseudo-code est qu'il permet d'exprimer presque en langage courant des algorithmes très précis. Cette description est, de plus, multi-cibles, c'est-à-dire qu'elle ne sous-entend pas l'utilisation d'un langage de programmation particulier. Des algorithmes élaborés en pseudo-code devraient alors être traduisibles dans beaucoup de langages de programmation courant.

Par ailleurs, le pseudo-code n'est pas un langage de programmation. Cette caractéristique le rend libre de toute contrainte qui serait liée à l'usage d'un compilateur.

3.3.8.4. Inconvénients du pseudo-code

Les erreurs commises lors de l'utilisation d'un tel langage ne sont pas facilement détectables. Si l'algorithme est important, le temps nécessaire à la détection des erreurs restera long.

Le fait que le pseudo-code soit multi-cibles peut se révéler être un inconvénient de taille. Parfois, la transposition d'un algorithme en langage structuré dans un langage de programmation précis nécessitera des changements plus que ponctuels. Dans des cas extrêmes, la structure de l'algorithme devra être revue dans son ensemble. Cette inconvénient majeur est issu du niveau d'abstraction adopté par le pseudo-code qui lui interdit de prendre en compte les contraintes du langage cible.

Pour contourner cet handicap, il serait judicieux d'adapter les concepts de base du langage structuré en fonction du langage cible. Malheureusement, une telle pratique éliminerait toute portabilité des algorithmes développés.

3.3.9. Les tables de décision

Les tables de décision constituent une technique de modélisation remarquable, utilisable dans la plupart des étapes du développement. D'après nos nombreuses discussions avec des développeurs, elles sont acceptées et reconnues par tous et, quoique vieilles de plus de trente ans, elles restent toujours d'actualité.

Les tables facilitent principalement la résolution de problèmes mettant en oeuvre un nombre élevé de combinaisons de conditions entraînant des actions variées. Les problèmes nécessitant des actions principalement séquentielles seront plutôt résolus à l'aide du langage structuré.

Leur principale différence par rapport à un algorithme est qu'elles ne sous-entendent pas l'emploi d'instructions définies. Excepté l'ordonnancement des règles qui nécessite un minimum de bon sens, aucune contrainte n'est imposée.

3.3.9.1. Présentation des tables de décision

Une table de décision résume une situation dans laquelle des actions doivent être prises en fonction des valeurs de certaines conditions.

Elle se compose de quatre parties :

A	B
C	D

Le quadrant A correspond à l'énoncé des conditions et le quadrant B à leurs valeurs.
Le quadrant C énonce les actions et le quadrant D leurs applications éventuelles.
Les quadrants B et D constituent les règles de la table, c'est-à-dire les actions à prendre pour chaque combinaison significative des valeurs des conditions.

Il existe plusieurs types de tables de décision. Nous ne reprendrons ci-dessous que ceux utilisés à la C.G.E.R.[27]. Le lecteur intéressé pourra lire [METZNER].

Les types de tables proposés par le M.A.I. sont :

- les tables à entrée limitée, où chaque condition ne peut prendre que deux valeurs.

- les tables à entrée étendue, où chaque condition peut prendre plus de deux valeurs

- les tables à entrée mixte, où l'on retrouve un mélange de condition à deux valeurs et à plus de deux valeurs.

La processus de construction d'une table comporte, selon le M.A.I. [MAI 7, chap. 2], cinq étapes.

- l'étape "texte" se résume à une lecture attentive de l'énoncé afin d'en dégager une liste brute des conditions et actions.

---

27. Des exemples de tables sont présentés en annexe I.

- l'étape "conditions" va permettre de déterminer toutes les conditions qui apparaîtront dans la table. Pour cela, on éliminera de la liste brute les redites et les contradiction. Ensuite, on construira la partie "condition" de la table en utilisant la règle de regroupement suivante : "regrouper les conditions de manière telle que :

  - les conditions dépendantes apparaissent ensemble, la plus dominante étant placée en premier lieu;

  - les conditions les plus variables, c'est-à-dire celles susceptibles de prendre le plus grand nombre de valeurs apparaissent en dernier lieu" [MAI 7, chap. 2, p.8].

On veillera également à ordonner les valeurs des conditions suivant un certain ordre (croissant pour les valeurs numériques, alphabétique pour les valeurs alphabétiques).

- l'étape "règles", lors de laquelle on transcrit les phrases lues dans le texte avec les conditions et actions définies. On utilisera exclusivement des (combinaisons de) conditions relevées dans l'étape précédente.

- L'étape de vérification de la table va permettre de contrôler son exhaustivité, sa non-contradiction, sa non-redondance et son exactitude.

On vérifiera notamment qu'il n'existe pas de règles de décision sans actions, que toutes les actions sont au moins exécutées dans une règle, et que les règles ne sont pas contradictoires deux à deux.

- L'étape d'optimisation de la table permettra principalement de réduire celle-ci par fusion de règles ayant les mêmes actions.

Enfin, il est possible de séparer une table trop volumineuse sur base d'un condition ou d'une action. On aura alors un réseau de tables ou la table initiale branchera successivement sur les tables secondaires.

3.3.9.2. Domaines d'application des tables de décision

Les tables de décision peuvent être employées à tous les niveaux du développement, depuis l'étude de l'existant jusqu'à la maintenance.

Lors de l'étude de l'existant et de l'analyse conceptuelle, elles vont fournir un moyen efficace pour décrire le système avec une extrême précision, obligeant les développeurs à éliminer toute imprécision.

Pour les analyses fonctionnelle et technique, les tables de décision vont être utilisées comme formalisme de description des traitements. Elles constitueront ainsi un langage de haut niveau qui sera aisément traduisible lors de la phase de programmation.

Les tables pourront également servir de support pour les tests du système. Les erreurs fonctionnelles seront rapidement détectées si une description du système sous la forme tables de décision existe.

Leur emploi dans la documentation du système facilitera grandement sa compréhension et permettra d'effectuer une maintenance rapide et efficace.

3.3.9.3. Avantages des tables de décision

La précision et la concision des tables de décision sont leur deux principaux avantages.

La précision est garantie par un processus de construction rigoureux et un caractère extrêmement précis des tables qui obligera les développeurs à réfléchir exhaustivement au problème posé.

La concision des tables est, quant à elle, remarquable. Celles-ci ont une forte capacité d'expression et de manipulation de conditions et d'actions nombreuses et complexes au sein d'une même table. En outre, elles regrouperont les informations pertinentes ou ayant des relations entre elles.

Les tables de décision peuvent être construites directement à partir d'un texte en langage courant. Elles ne nécessitent aucun autre formalisme intermédiaire. De ce fait, les développeurs pourront directement travailler sur des spécifications normalement correctes. Le résultat sera dès lors plus fiable.

Enfin, un dernier avantage réside dans le fait que, si elles sont employées tout au long du développement, un seul formalisme sera nécessaire pour décrire le système. La compréhension de ce dernier, et sans doute sa qualité, n'en seront qu'améliorées.

3.3.9.4. Inconvénients des tables de décision

Un inconvénient des tables de décision est leur lourdeur de construction lorsque les conditions sont très variées. Il est, en effet, nécessaire de construire d'abord une table complète avant de l'optimiser, et ce, afin d'éviter toute erreur. Comme aucun outil aidant à la construction des tables de décision n'existe à notre connaissance, l'élaboration peut devenir fastidieuse, voire un facteur de rejet.

Un second inconvénient des tables de décision, qui est plus une limite, est leur caractère non-procédural.

Cette limite réduit, à notre avis, leur possibilité d'application dans la description des programmes. Celle-ci doit en effet être telle qu'aucune décision ne sera prise lors de la phase de codage. Or, l'ordonnancement des actions n'est pas toujours aléatoire. Il serait donc utile de préciser cet ordre ou d'éviter l'emploi des tables dans de pareils cas.

3.3.10. Jackson Structured Programming

La présentation ci-dessous trouve ses sources dans un travail de synthèse réalisé par Etienne de GRADY [GRADY, p.4 et suivantes]. Cette synthèse présente la méthode de JACKSON de façon générale en s'inspirant des travaux de JACKSON lui-même, mais aussi d'autres auteurs ayant réalisé des études de cette méthode (notamment [MAT]).

3.3.10.1. Présentation succincte de la méthode

Le principe de base de la méthode est la décomposition des structures de données du problème et leur mise en correspondance. Ces structures de données, mais aussi leurs correspondances, serviront de guides lors de l'élaboration des structures des programmes.

Le processus d'élaboration d'un programme peut se résumer en trois étapes.

Tout d'abord, on identifiera les structures de données en entrée et en sortie et on en donnera une représentation sous forme arborescente. Pour réaliser celle-ci, trois composants de base sont proposés : la séquence, la sélection et l'itération[28].
Ensuite, on analysera les deux structures de manière descendante afin de découvrir les parallélismes entre les niveaux correspondants.

---

28. Cfr. annexe J.

Enfin, la structure de l'algorithme sera directement issue de la comparaison faite entre les entrées et les sorties et utilisera également les trois éléments de base[29].

Le type de programmes abordables avec cette méthode reçoivent en entrée un ou plusieurs flux de données séquentiels. Quoique initialement destiné aux programmes batch, cette méthode pourrait également être utilisée pour des programmes "on-line". Nous n'entrerons cependant pas ici dans une étude d'applicabilité de J.S.P. à ce type de programmes. Là n'est pas notre but.

Des problèmes de conflits de structures entre entrées et sorties peuvent parfois se produire. Des modifications seront alors envisagées afin de rendre ces structures semblables.

3.3.10.2. Domaines d'application de J.S.P.

La méthode proposée par JACKSON est surtout destinée à décrire les programmes. Son domaine d'application privilégié est donc l'analyse technique.

L'utilisation d'une partie seulement de la méthode, à savoir la structuration des données et des résultats, peut s'avérer être une bonne façon d'exprimer des spécifications. Elles seraient alors complétées par l'adjonction de contraintes sur les relations existantes entre les données et les résultats.

Par ailleurs, cette méthode peut être utilisée pour décrire rapidement et simplement les structures des données tout au long de l'analyse.

---

29. Cfr. annexe J.

3.3.10.3. Avantages de J.S.P.

J.S.P. permet de structurer directement un algorithme à partir des données et des résultats. Si les flux en entrée et en sortie sont séquentiels, la réalisation d'un programme ne sera qu'un jeu d'enfant.

3.3.10.4. Inconvénients de J.S.P.

La méthode J.S.P. est relativement floue. Les règles qu'elle préconise pour décomposer les structures de données ne conduisent pas toujours aux mêmes résultats. Or, si une structure de données peut être représentée de diverses manières, généralement une seule d'entre elle aboutira à la construction d'un programme valable.

Le formalisme initialement introduit par JACKSON était destiné à construire des programmes "batch", ne recevant en entrée qu'un flux séquentiel, et ne produisant qu'un flux séquentiel. Des adaptations ont été réalisées afin d'étendre le domaine d'application aux programmes recevant plusieurs flux séquentiels en entrée.

En ce qui concerne les programmes "on-line", il serait également possible d'utiliser J.S.P. N'ayant personnellement pas pu effectuer des tests dans ce domaine, nous n'oserons pas réfuter cette affirmation. Signalons toutefois que les développeurs que nous avons rencontrés étaient hostiles à l'idée d'utiliser J.S.P. (soi-disant démodé) pour de tels programmes.

3.4. Les outils d'aide au développement de logiciels

3.4.1. Le dictionnaire de données DataManager

"Un dictionnaire de données est un système d'informations composé :

- d'une base de données dont les éléments contiennent des informations au sujet des 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 toutes les 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" [DEDE, p.62].

Le dictionnaire de données DataManager (de la firme anglaise Management and Systems Programming) est utilisé à la C.G.E.R. depuis plusieurs années.

Un des buts principaux d'un tel dictionnaire est la centralisation de la documentation relative aux systèmes d'information. En effet, ces systèmes sont souvent l'oeuvre de multiples personnes ayant parfois des vues différentes de la réalité. La centralisation va permettre d'évincer toute vision erronée du système pour laisser place à une seule définition acceptée par tous (ou presque).

Pour atteindre ce but de documentation (la documentation est d'une importance capitale pour l'entreprise, que ce

soit pour contrôler l'état d'avancement des projets, ou que ce soit pour assurer une maintenance correcte), il importe de remplir correctement ce dictionnaire avec des renseignements adéquats et fiables.

Des adaptations ont été réalisées en fonction des besoins de la Caisse d'Epargne. Celles-ci se caractérisent par l'ajout de nouveaux types de membres définissant plus finement les éléments introduits dans le dictionnaire.

Nous allons nous limiter ici à une présentation réduite de DataManager, mais nous nous attarderons plus longuement sur les capacités réelles d'un tel dictionnaire. Le lecteur intéressé par l'utilisation de DataManager pourra lire [LUYCKX].

3.4.1.1. Présentation de DataManager

Le logiciel de base définit six types d'entités (ou types de membres) acceptables par sa base de données. Néanmoins, il est possible de créer d'autres sous-types et de les ajouter dans la définition de la base.

Les six types de base sont, pour les données, les FILE, GROUP et ITEM, et, pour les traitements, les SYSTEM, PROGRAM et MODULE. La configuration de DataManager adaptée pour la C.G.E.R. est reprise sur les schémas en annexe K.

a) Le langage de définition des données

Lorsque l'utilisateur introduit des membres - appartenant à un type de membres- dans le dictionnaire, il a à sa disposition des clauses lui permettant de définir ce nouveau membre.

Les *clauses descriptives*, composées de un ou plusieurs mots-clés ainsi que de zones variables à remplir, donneront les caractéristiques du membre introduit.

Parmi les clauses descriptives, en voici quelques unes, communes à tous les types de membres.

- DESCRIPTION : Cette clause permet à l'utilisateur de décrire le membre comme il le désire.

- ALIAS : Cette clause, qui peut être répétitive et qui doit au moins figurer une fois dans la description, associe au membre un ou plusieurs noms alternatifs. Cette facilité est très appréciée car elle autorise le bilinguisme dans le dictionnaire (ALIAS français et ALIAS néerlandais).

- CATALOGUE : Cette clause offre la possibilité d'affecter plusieurs mots-clés au membre. Ceux-ci permettront facilement de retrouver un membre dont on a oublié le nom. De plus, ils définissent implicitement des familles auxquelles sont rattachés ces membres.

Les *clauses relationnelles* permettent d'introduire des relations entre membres de types de membres. Les relations autorisées sont réglementées. Ainsi, pour un membre "a", de type A, rattaché à un membre "b", DataManager définit l'ensemble des types acceptables pour le membre "b", qui lui permettent d'être en relation avec "a".

b) Le langage de manipulation des données

Au-delà des clauses, l'utilisateur a également à sa disposition un langage de manipulation des données. L'exposé détaillé de ce langage dépasse le cadre de

cette étude. Aussi nous bornerons-nous à citer les différentes possibilités de ce langage[30].

Les *commandes de manipulation de données* permettent, entre autres, d'insérer, de modifier, de supprimer des membres dans la base de données. L'ensemble des commandes disponibles est plus vaste. Mais, par soucis de clarté, nous n'entrerons pas dans sa description complète.

Les *commandes d'interrogation de données* permettent, via l'édition de rapports, de connaître les différentes relations existant entre les membres. Elles donnent aussi la possibilité de travailler d'après les mots-clés et les ALIAS.

Au delà de ces commandes, DataManager offre également des commandes auxiliaires (permettant par exemple de conserver des données en mémoire centrale) ainsi qu'un système de sécurité.

3.4.1.2. Les avantages de DataManager

DataManager est un outil de documentation remarquable qui permet, grâce à la richesse de ses structures de représentation et s'il est bien utilisé, de centraliser toutes les informations pertinentes relatives aux systèmes d'informations développés.

Cette centralisation permet d'avoir une seule bonne référence contenant toutes les données relatives à tous les systèmes développés ou en cours de développement.

De plus, sa structure permet l'adjonction de programmes permettant d'effectuer des requêtes

---

30. Une étude plus complète a été réalisée dans [DEDE, p.80 et suivantes].

sophistiquées (par exemple, reconstituer l'arborescence d'un système) pouvant produire des résultats très utiles pour les développeurs, mais aussi pour d'autres services de gestion.

La capacité d'évolution dans le temps de ce dictionnaire lui permet de rester toujours opérationnel et ce, même après plusieurs années d'utilisation.

3.4.1.3. Les inconvénients de DataManager

Le principal point négatif concernant ce dictionnaire est sa lourdeur d'utilisation. Lors de nos nombreuses discussions avec les personnes l'utilisant (ou qui devraient l'utiliser), ce désavantage était chaque fois mentionné en premier lieu.

Cette lourdeur est causée par l'absence d'interface utilisateur conviviale. Et, pour peu que l'utilisateur de DataManager soit contre l'idée de l'employer, cette interface devient tout à fait hostile. Lors de différents essais, nous avons pu, en effet, constater la rudesse du dialogue homme - machine de ce dictionnaire.

Il en découle qu'il n'est bien souvent utilisé que pour pouvoir exécuter PALOMA. Pour les autres étapes du développement, il est tout simplement ignoré.

Nous pensons qu'il est dommage qu'un outil d'une telle puissance soit si peu utilisé à cause de son interface. La mise au point d'une nouvelle interface plus performante, orientée utilisateur au maximum, augmenterait sans nul doute le taux d'utilisation de DataManager. La qualité des produits développés et leur facilité de maintenance n'en seraient que meilleures.

Son autre faiblesse est la manque d'interfaces disponibles avec d'autres logiciels, notamment EXCELERATOR[31]. Comme nous le verrons, EXCELERATOR rend enfantine la construction de modèles conceptuels. Malheureusement, les données introduites dans ce logiciel sont, à l'heure actuelle, inutilisables par DataManager.

L'étude d'une interface entre les deux produits serait elle aussi très pertinente et adoucirait l'humeur des développeurs amenés à introduire deux fois les mêmes données.

Un dernier point noir du dictionnaire est son impossibilité des générer automatiquement des éléments à partir de son contenu.

3.4.2. PALOMA

PALOMA (PAth LOad MAtrix) est un logiciel développé à la C.G.E.R. destiné à fournir toutes les informations nécessaires pour la construction du modèle dynamique des données. De plus, ce logiciel valide les modèles conceptuels des données et des traitements par une confrontation mutuelle. Enfin, PALOMA estime la charge représentée par les tâches du système.

La philosophie de la méthode de développement initiale introduite par la M.A.I. reste présente dans PALOMA qui est très orienté "données" et base de données :

1°- PALOMA sert de base à l'élaboration du modèle dynamique des données.

---

31. Cfr. infra 3.4.3.1.

2°- PALOMA ne considère dans le modèle conceptuel des données que les entités et les associations porteuses. Cette approche est très orientée base de données et ne fournira au D.B.A que les informations nécessaires à l'élaboration du modèle logique des données.

Nous verrons dans la critique de PALOMA qu'il n'est pas toujours précis, et que son emploi actuel pourrait être optimisé, même si l'optique initiale n'est plus respectée.

Signalons également que PALOMA repose sur le dictionnaire de données DataManager décrit ci-dessus. Ces deux outils sont fortement liés et renforcent leurs positions de force dans la méthodologie de développement de la C.G.E.R.

3.4.2.1. Les buts de PALOMA

PALOMA a été conçu pour être exécuté en fin d'analyse conceptuelle et doit fournir des estimations brutes destinées à orienter l'analyse fonctionnelle. Son but n'est nullement de calculer suivant des modèles compliqués et extrêmement précis les charges et accès. Une telle exactitude serait d'ailleurs impossible au niveau de l'analyse conceptuelle.

En résumé, PALOMA est destiné à fournir un bénéfice maximum pour un investissement minimum.

3.4.2.2. Présentation de PALOMA

a) Résultats fournis par PALOMA

Au niveau du modèle conceptuel des données, PALOMA va estimer la charge supportée par chacun des arcs qui relient les divers objets du modèle.

Cette "estimation du trafic sur un arc tient compte:

- du nombre de tâches qui le parcourent;
- du nombre maximum d'exécutions quotidiennes de ces tâches
- et du mode d'accès à l'extrémité de l'arc : lecture, modification, insertion et suppression" [MAI 8, p.3].

Au niveau des tâches, "PALOMA estime la charge représentée (...) [par chacune d'elles], sur base du parcours qu'elles effectuent dans le M.C.D., du nombre d'occurrences d'objets mises en oeuvre, et de la manière dont on accède à ces objets (lecture, modification, insertion ou suppression)" [MAI 8, p.3].

Remarquons que les tâches on-line et les tâches batch font l'objet de traitements séparés.

Les informations fournies par le logiciel sont :

- une liste récapitulative des charges des arcs et des points d'accès;
- une liste détaillée par arc et par point d'accès de leur charge;
- une liste condensée résumant la charge des tâches;
- une liste détaillée explicitant le parcours de chaque tâche;
- un listing d'erreurs.

b) Informations utilisées par PALOMA

Pour chaque objet appartenant au modèle conceptuel (rappelons que ne sont considérées comme objets que les entités et associations porteuses) il suffit d'introduire le nombre d'occurrences et les cardinalités des liens existants entre cet objets et les autres objets du modèle.

Pour les tâches importantes, manipulant des volumes considérables de données et/ou fréquemment utilisées, on indique le type de la tâche, le nombre d'exécutions quotidiennes et le parcours effectué dans le modèle conceptuel des données.

3.4.2.3. Avantages de PALOMA

Le logiciel développé à la C.G.E.R. présente toute une série d'avantages appréciables.

Tout d'abord, comme c'est un logiciel "maison", il s'intègre parfaitement dans le processus de développement déjà en place. Il utilise des modèles élaborés en amont et fournit de informations précieuses pour les modèles qui seront produits en aval.

PALOMA repose aussi sur l'utilisation du dictionnaire de données DataManager, dont l'emploi est fortement recommandé et encouragé à la Caisse d'Epargne.

Ensuite, pour peu que l'on surmonte les quelques difficultés d'utilisation de DataManager, PALOMA n'exige l'introduction que d'un minimum de données pour effectuer ses calculs. De plus, il offre des facilités par l'insertion automatique de valeurs par défaut les plus probables, allégeant ainsi grandement le travail d'introduction des données.

Un autre avantage du logiciel est sa capacité de signaler les anomalies qui figurent dans le modèle Entité-Association. Il suggérera en effet via des "warnings" les associations qui, à son sens, manquent dans le schéma. Par exemple, si, lors de l'estimation du trafic, PALOMA constate qu'il existe un accès fréquent à un objet B au départ d'un objet A, et qu'ils ne sont pas directement reliés, il produira un

"warning" déclarant qu'une association entre A et B serait peut-être nécessaire.

De même, via cette estimation des trafics et surtout des points d'entrée dans le modèle, PALOMA va aider considérablement le D.B.A. dans son travail de transformation du modèle conceptuel en un modèle logique. Cette aide est encore accentuée par l'orientation "données" de PALOMA dont nous avons discuté plus haut.

Enfin, les traitements pourront aussi être revus si les estimations fournies par le logiciel mettent en évidence des tâches trop lourdes ou inexécutables.

3.4.2.4. Inconvénients de PALOMA

Malgré tous les avantages qu'il offre, le logiciel comporte certains inconvénients. Ceux-ci se situent principalement au niveau des calculs effectués et de leur précision.

Comme nous l'avons vu dans la présentation du produit, PALOMA fait des estimations de charge des arcs du modèle conceptuel des données en tenant compte du mode d'accès (écriture, insertion, modification ou suppression). Ces pondérations en fonction du mode d'accès sont établies par le D.B.A. Cependant, après plusieurs discussions sur ce sujet, il est apparu qu'elles n'étaient pas toujours très correctes et que, dès lors, les résultats pouvaient s'en ressentir.

Un autre facteur influençant les résultats est le nombre de tâches prises en compte pour exécuter PALOMA. Il est recommandé de n'introduire que celles qui sont significatives. Mais une subjectivité inévitable entrera en ligne de compte pour la sélection de ces tâches significatives. De plus, une évolution dans l'utilisation du système pourrait

augmenter le taux d'utilisation de certaines tâches tout en diminuant celui d'autres. Les résultats initiaux de PALOMA, transcrits sur un modèle dynamique des données, pourraient donc conduire à l'élaboration d'un schéma de base de données qui se révélerait moins performant que prévu.

Lors de nos entretiens avec le D.B.A., il est apparu que la première exécution de PALOMA ne fournissait que des résultats approximatifs. Par contre, elle permettrait de modifier éventuellement le modèle conceptuel des données initial en tenant compte des avertissements. Une seconde exécution sur base de ce nouveau modèle produirait alors des résultats nettement plus précis.

Malheureusement, une seule itération n'est autorisée. Les souhaits d'utilisation en cascade, notamment formulés par le D.B.A., sont donc actuellement vains.

3.4.3. Les outils C.A.S.E.

Comme nous l'avons déjà signalé maintes fois, le développement d'un système d'information moderne est le fruit d'un travail de longue haleine touchant une variété importante de personnes.

Ces différentes personnes seront habituellement impliquées dans un stade seulement du développement.

Si l'on considère le système d'un point de vue abstrait comme un "objet" modifié au cours du processus de développement (exemple : au niveau conceptuel, le système se compose des différents modèles le décrivant; au niveau de la programmation, le système est constitué de l'ensemble des programmes et de toute la documentation le concernant), on peut raisonnablement penser que, si un minimum de soin n'est pas apporté au cours de chaque étape du développement, le système final

ne reflétera que sommairement le système initialement conçu au tout début de l'analyse.

Les outils C.A.S.E. (Computer Aided Software Engineering) apportent une réponse à ce genre de problèmes. Leur idée sous-jacente est de profiter au maximum des spécifications initiales du système en minimisant les interventions humaines ou en les simplifiant au maximum. Il est en effet bien connu que les développeurs désapprouvent de faire plusieurs fois la même chose. Or, spécifier un logiciel ou en réaliser les programmes sont deux choses similaires, mais dans des "langages" différents. Les outils C.A.S.E. vont éviter ce dédoublement du travail en récupérant, pour chaque niveau d'analyse, les spécifications du niveau précédent. Grâce à cette d'approche, l'hostilité des concepteurs devrait disparaître

Leur utilisation augmenterait donc la productivité des développeurs et la qualité des systèmes produits en atténuant ainsi la crise du logiciel.

Les meilleurs outils C.A.S.E. devraient être capables de produire des programmes exécutables directement à partir des spécifications du système. Pour atteindre ce but, une démarche des plus rigoureuses doit être suivie et une documentation importante doit exister. En fait, ces logiciels vont forcer leurs utilisateurs à réfléchir dès le début de l'analyse et à ne plus bâcler les étapes conceptuelles et fonctionnelles.

Cependant, la démarche devra quand même permettre des adaptations humaines des résultats obtenus. Celles-ci devraient toutefois n'être que ponctuelles et ne pas remettre en cause la solution.

La documentation, quant à elle, garde toujours une importance capitale. Elle est indispensable à la maintenance inévitable des systèmes d'information modernes. Par ailleurs, nous pensons que cette

documentation est d'autant plus indispensable qu'une partie du travail est prise en charge par les outils C.A.S.E., donc cachée à l'informaticien. La documentation permettra à l'organisation de se souvenir des choix qui ont été effectués, des critères les ayant guidés et des adaptations humaines réalisées.

De plus, il faut éviter de tomber dans le travers de la technologie en enlevant son savoir et son savoir-faire à l'homme pour les donner à la machine. Cette dernière ne sera jamais qu'une exécutante, et elle aura toujours besoin de spécifications de plus en plus précises pour produire des logiciels de qualité.

Mais, une réflexion philosophique sur ce sujet n'étant pas l'objet de ce paragraphe, nous allons revenir à une étude plus terre-à-terre des outils d'automatisation du développement.

La littérature actuelle concernant les outils C.A.S.E. étant peu étoffée, nous allons uniquement en présenter deux que nous avons eu l'occasion de côtoyer lors de notre séjour au M.A.I.

Ces deux outils sont présentés par leurs démonstrateurs respectifs comme complémentaires. Le premier, EXCELERATOR, servira surtout au début du développement alors que le second, TELON, entrera en action dans les étapes ultérieures.

3.4.3.1. EXCELERATOR

EXCELERATOR est un logiciel développé par la société américaine Index Technology Corporation. Il est apparu sur le marché mondial au milieu de l'année 1984.

Ce logiciel est actuellement présent en Belgique sur plus de cent sites différents (un site correspond à un

poste de travail, soit un PC), et serait numéro un en France où il servirait de base à la méthode Merise.

Il a été conçu de manière à pouvoir supporter différentes méthodes ainsi que leurs symbolismes respectifs et il constitue surtout un outil d'aide pour le début de l'analyse (analyse conceptuelle, exigences et limites du système, début d'analyse fonctionnelle, mais aussi l'étude de l'existant et les phases de planning). D'après ses démonstrateurs, EXCELERATOR a pour but d'accélérer le développement des logiciels en ayant recours à une technologie évoluée et en constante évolution, en offrant des fonctionnalités de plus en plus variées, et en intégrant au maximum le développement.

Avant d'entrer dans la description du produit, signalons que nous avons eu l'opportunité de l'utiliser. Et, quoique nous n'ayons pas pu les explorer à fond, ses possibilités nous ont paru remarquables. Les qualités graphiques de ce logiciel sont d'un niveau élevé et son dictionnaire de données nous a semblé satisfaisant. D'un autre côté, nous n'avons malheureusement pas pu nous procurer toute la documentation requise pour une présentation des concepts de base du système. Aussi nous limiterons-nous à présenter les différents modules du logiciel.

a) Les possibilités d'EXCELERATOR

Le produit complet se compose de sept modules.

Le module *DICTIONARY* est le coeur même du logiciel. Il permet le stockage de toutes les informations concernant le nouveau système (les structures de données, les définitions des écrans et des rapports, etc.) à un seul endroit.

Le dictionnaire enregistre automatiquement les relations logiques entre les données. De plus, il est capable de créer et d'analyser des références croisées ("where used" et "used by"). Enfin, il peut vérifier certaines contraintes sur les données (par exemple la non redondance) en utilisant les concepts de normalisation.

Le module *HOUSEKEEPING* est constitué d'un ensemble d'utilitaires variés permettant de (re)configurer le système, d'archiver un ou plusieurs projets ou de récupérer des projets archivés, et de gérer le système, c'est-à-dire d'ajouter, de modifier ou de supprimer des utilisateurs, des projets, des mots de passe, ou des privilèges d'accès.

Le module *GRAPHICS* est celui avec lequel nous avons le plus travaillé. Il permet de construire six types de schémas différents. C'est notamment cette variété qui autorise l'emploi du logiciel avec plusieurs méthodes. De plus, il est possible de demander une adaptation en fonction d'une méthode ayant un formalisme non repris dans ce module.

Les six types de schémas autorisés sont :

- Les diagrammes de flux de données;
- Les "structure charts";
- Les "structure diagrams";
- Les modèles logiques des données;
- Les schémas Entité-Association;
- Les "presentation graphs".

La facilité de manipulation des objets de ces différents schémas est assez impressionnante. Par exemple, pour raccorder deux entités ensemble via une association, trois "clics" de la souris suffisent. Un habitué (mais non expert) devrait raisonnablement parvenir à construire un schéma en

un temps minime (5 à 10 minutes pour un E-A d'une quinzaine d'objets).

Le module *SCREENS & REPORTS* offre la possibilité de constituer des prototypes des futurs états (écrans et imprimés) qui serviront de support à la communication homme - machine du nouveau système. L'emploi de la souris permet de les concevoir en un minimum de temps. Cette rapidité d'exécution favorise de surcroît la participation de l'utilisateur à leur élaboration.

Le module *EXTENDED ANALYSIS* est un module de contrôle du travail effectué. A partir des schémas élaborés et des données introduites dans le dictionnaire, des contrôles de complétude, de cohérence et de précision pourront être menés. Il est par exemple possible de vérifier l'état de normalisation d'un schéma.

Le module *INTERFACE* autorise l'élaboration d'un projet par plusieurs personnes travaillant sur des stations différentes. Ce module collecte toutes les informations qu'il centralise sur un "host". Il gère tous les problèmes d'accès aux données, de privilèges, de statistique (qui a fait quoi), etc.

En outre, cette interface est capable de gérer les transferts de données avec d'autre logiciels de développement.

Le module *DOCUMENTATION* offre la possibilité aux développeurs d'élaborer des documents en utilisant directement des données du dictionnaire et de fichiers de traitements de texte. La mise au point de pareils documents est supportée par une table des matières graphique qui visualise leur structure.

b) Les avantages d'EXCELERATOR

Ce qui nous a le plus frappé lors de nos contacts avec le logiciel est la vitesse avec laquelle on parvient à élaborer des schémas d'un système. En un minimum de temps, il est possible d'apprendre les fonctionnalités les plus importantes et d'être déjà productif. Dès lors, il est concevable que des utilisateurs développent eux-même certaines parties du futur système les touchant plus particulièrement. Nous pensons par exemple aux formats des écrans et des rapports.
L'alternative est également possible et serait basée sur une collaboration étroite entre développeurs et utilisateurs finaux mettant ensemble au point ces différents états.

c) Les inconvénients d'EXCELERATOR

Une des faiblesses de ce logiciel est la qualité médiocre de l'interface qu'il propose avec d'autres outils de développement.

Le problème que nous avons rencontré empêche actuellement une utilisation optimale d'EXCELERATOR. Le dictionnaire utilisé à la C.G.E.R. est DataManager. Toutes les informations concernant les systèmes d'informations sont actuellement contenue dans ce dictionnaire. Malheureusement, la seule interface proposée avec EXCELERATOR ne supporte que le transfert des données d'un dictionnaire à l'autre. Aucun contrôle n'est effectué. L'installation d'une telle interface serait très dangereuse et occasionnerait sans doute pas mal de dégâts. Plus aucun contrôle ne serait possible et les informations reprises dans DataManager ne pourraient plus être considérées comme sûres.

Ce problème d'interfaçage entre plusieurs produits subsistera tant qu'aucune normalisation reconnue internationalement ne sera pas établie.

Cependant, on pourrait envisager d'utiliser EXCELERATOR comme outil de base sur lequel on grefferait d'autres outils, par analogie à un système d'exploitation grâce auquel il est possible d'utiliser d'innombrables applications. Cette possibilité n'est, selon nous, concrétisable que si EXCELERATOR est le premier outil utilisé par l'organisation.

Une critique que nous portons vis-à-vis de la présentation du produit par le producteur est qu'il n'explique pas clairement quels sont les concepts sous-jacents aux différents modules. Au lieu de se limiter à la présentation de ces derniers, il serait, à notre avis, plus intéressant de présenter d'abord des notions telles que celles de fichier, d'utilisateur, de projet, d'accès, etc., puis d'expliquer comment ces concepts sont mis en oeuvre au travers des divers modules.

3.4.3.2. TELON

TELON est un logiciel d'aide au développement élaboré par la Pansophic Systems Inc. D'après ses représentants et les représentants d'EXCELERATOR, il constitue le complément possible de ce dernier et termine le cycle d'analyse d'un système. Notons toutefois que TELON peut être employé comme seul outil C.A.S.E.

TELON couvre les phases de construction et d'implémentation de la solution, ainsi que la phase d'installation de l'implémentation.

Il a été conçu de manière à pouvoir s'adapter au plus grand nombre d'environnements différents, et de supporter des modifications de configuration. Pour aider cette intégration à un matériel déjà existant, TELON est implémenté différemment selon l'environnement.

a) Présentation de TELON

TELON est composé de dix modules différents qui seront ou non intégrés lors de l'installation sur un site particulier.

La *TELON DIRECTORY* est le centre nerveux du logiciel où toutes les informations concernant les applications sont stockées. Ce module joue le rôle de dictionnaire de données.

Le module *DATA ADMINISTRATION* gère les accès aux données dans les différents S.G.B.D. Toutes les autorisations sont définies dans ce module qui permet en outre l'utilisation de plusieurs bases de données pour une application.

L'*ANALYSIS TOOL INTERFACE* procure une interface autorisant la communication avec certains autres outils C.A.S.E. afin de travailler efficacement avec eux (par exemple avec EXCELERATOR).

Le *DATA DICTIONARY INTERFACE* permet de transférer des données présentes dans certains autres dictionnaires (par exemple le dictionnaire d'EXCELERATOR) vers celui de TELON afin d'éviter de devoir les ré-entrer.

Le *SCREEN/REPORT PAINTING* donne la possibilité de développer des écrans et des rapports interactivement. Ce module est proche du module SCREEN & REPORT d'EXCELERATOR.

Le module de *PROTOTYPING* permet de construire des prototypes du futur système et d'ainsi simuler son exécution devant l'utilisateur final. Cette exécution ne requiert aucune compilation et peut être extrêmement rapide.

Le prototyping est surtout utilisé pour mettre au point les enchaînements d'écrans et spécifier leur contenu.

Le module *SPECIFICATION FACILITY* permet aux développeurs d'introduire des spécifications variées dans un langage non procédural.

Le *TELON GENERATOR* génère, à partir des spécifications, un programme exécutable (en COBOL, COBOL II ou PL/1, avec accès possibles à des bases de données DB2, IMS/DB, VSAM ou à des fichiers séquentiels) bien structuré, propre, portable et documenté.

Le *TELON TEST FACILITY* offre des capacités intéressantes de test des programmes. Les développeurs testent le nouveau programme et corrigent les erreurs de spécification instantanément. En outre, ce module permet le traçage des accès aux données. Ce traçage peut être sélectif, c'est-à-dire que le testeur peut demander le suivi de certaines données uniquement. Cette possibilité peut, à l'occasion, accélérer la correction.

L'*AUTOMATED DOCUMENTATION* produit des rapports sur les différentes relations entre les données du dictionnaire. De plus, des rapports spécifiques indiqueront quels programmes risquent d'être affectés par le changement d'un schéma de base de données, ou encore quelles bases de données sont utilisées par quels programmes.

b) Remarques concernant TELON

Malgré nos nombreux déplacements, il nous fut impossible d'assister à une démonstration complète de TELON. Les seuls modules que nous avons vu sommairement en action sont ceux de "screen & report design" et de "prototyping".

Le "design" des écrans se déroule rapidement et ne requiert aucune connaissance particulière, si ce n'est les commandes de base de l'éditeur. Une fois les écrans terminés, le prototypage indiquera si rien n'a été omis et il permettra d'évaluer le niveau de satisfaction des utilisateurs.

Remarquons que, n'ayant recours à aucune compilation, le "design" et le prototypage peuvent se dérouler tout à fait simultanément. Après chaque écran, il est possible de recommencer le prototypage et de modifier tout au point par point. Cette propriété de TELON peut influence positivement la qualité du résultat qui deviendra vraiment du "sur mesure".

Cette facilité d'utilisation favorisant le prototypage risque de se retourner contre les développeurs peu soigneux. Si rien de précis n'existe au départ, l'approche essais-erreurs risque d'aboutir à une solution ne correspondant absolument pas aux besoins réels et il sera souvent indispensable de tout recommencer.

En ce qui concerne les autres modules, nous n'avons eu aucune opportunité de les tester. Nous ne pouvons dès lors porter un quelconque jugement sur leurs qualités ou leurs défauts.

3.4.3.3. Synthèse critique des outils C.A.S.E.

Nous n'avons pas la compétence pour réaliser une critique générale des outils C.A.S.E. Néanmoins, il nous semble qu'il serait judicieux d'insister sur les causes de leur apparition et sur leurs dangers.

Les causes, tout d'abord, ont déjà été annoncées plus haut. Ce sont, premièrement, le manque de productivité des développeurs et, deuxièmement, le coût sans cesse croissant de la maintenance.

Les dangers de tels outils, et plus particulièrement des générateurs de code résident dans l'aspect caché d'une partie plus ou moins importante du processus de développement. Si les générateurs sont mal utilisés, l'effet opposé de celui escompté (augmentation de la productivité et diminution du coût de la maintenance) se produira, à savoir une augmentation des coûts de maintenance et une diminution de la productivité.

Par exemple, si aucune documentation n'est rigoureusement mise-à-jour lors de modifications manuelles ou automatiques des programmes -et de telles modifications sont inévitables-, il ne sera plus possible de maîtriser un produit qui se comportera tout à fait aléatoirement.

Ces deux derniers points nous amènent à douter de la réelle efficacité des outils C.A.S.E.

Reprenons tout d'abord la considération concernant la mauvaise utilisation éventuelle d'un générateur. Celle-ci n'est, à notre avis, possible que si les développeurs utilisent mal le langage de spécification de haut niveau, ou qu'ils accordent au logiciel une intelligence supérieure à celle qu'il possède réellement (ils s'imaginent alors que le générateur est capable de résoudre des problèmes à leur place).

Une utilisation optimale de l'outil présupposerait alors une connaissance de ses mécanismes de fonctionnement internes, tout comme pour le langage PROLOG. Nous serions alors en présence de développeurs à deux vitesses. Nous aurions d'un côté ceux qui connaissent bien le fonctionnement du générateur et qui l'exploitent au mieux et, de l'autre côté, les "ignorants" qui l'utiliseraient anarchiquement.

Le but des générateurs, à savoir la production d'un programme exécutable à partir de spécifications de haut niveau introduites par un utilisateur, ne peut dès lors pas être considéré comme atteint. En effet, dès l'instant où une connaissance du fonctionnement de l'outil est requise, il n'est plus possible de considérer les spécifications autrement qu'un langage de programmation de plus haut niveau.

En ce qui concerne les modifications que les développeurs peuvent apporter aux programmes générés, elles introduisent également un doute quant à la véritable efficacité des générateurs de code. Si elles permettent d'améliorer la justesse et les performances du programme, c'est parce que le générateur est incapable de réaliser lui-même un programme tout à fait efficace.

Un compromis possible consisterait à accepter ces programmes moins efficaces qui, grâce à l'évolution constante du matériel, seraient quand même aussi puissants que les programmes d'antan.

Un autre danger des outils C.A.S.E. est qu'ils risquent d'enlever leur compétence aux développeurs en employant des techniques de travail de plus en plus sophistiquées. Ils deviendraient alors tout puissants face à des développeurs dont l'ignorance serait croissante.

Un dernier aspect sur lequel nous aimerions clore cette petite analyse des outils C.A.S.E. est la différence qui les sépare des traditionnels outils de C.A.O. (Conception Assistée par Ordinateur).

"Contrairement aux outils de C.A.O., les outils C.A.S.E. actuels ne possèdent aucune connaissance sémantique du domaine dans lequel ils sont utilisés. (...) La logique propre à un domaine de gestion doit être 'codée' complètement par l'utilisateur" [DATA].

En considérant cet aspect, on peut aisément deviner quel serait le gain de productivité si les développeurs avaient à leur disposition une base de connaissance se rapportant au domaine qu'ils étudient.

Malheureusement, tel n'est pas encore le cas. Toute la logique reste encore le fruit de la réflexion des développeurs. Il leur faudra donc considérer les outils C.A.S.E. uniquement comme des supports et non comme des "assistants" capable de penser à leur place.

3.5. Classification des modèles et outils.

Nous allons à présent tenter de classer les modèles et outils présentés dans les différentes étapes du développement.

Cette classification se fera à deux niveaux. Pour chaque étape du développement, nous allons reprendre les modèles et outils qui pourraient apporter une aide aux concepteurs. Nous proposerons ensuite un classement au sein de l'étape, du plus approprié au moins utile.

Pour réaliser cette classification, nous avons pris en compte les avantages et inconvénients que nous avons identifiés.

3.5.1. Le niveau conceptuel

Du côté des données, nous avons principalement le modèle Entité-Association. Son utilisation est, à notre avis, quasi indispensable lors du développement d'un système. Le symbolisme utilisé peut varier légèrement, mais les principes de base restent les mêmes. L'amélioration de la perception de réel et de la communication peut être considérable. Nous classons donc ce modèle parmi les "must".

D'autre part, l'extension de ce modèle par l'adjonction des volumes -conduisant à un modèle statique des données- peut se révéler intéressante si ces volumes sont suffisamment précis et bien exploités.

Du côté des traitements, le D.F.D. figure au premier plan. Il a l'avantage de regrouper en son sein la statique et la dynamique du système. De plus, il visualise les flux d'informations circulant au travers des différentes tâches. Un D.F.D. bien construit peut s'avérer très utile pour le reste de l'analyse.

Vient ensuite la S.H.A.T. Ce modèle est également très utile, mais son défaut principal reste sa lourdeur en cas de gros système. Néanmoins, son utilisation pour la décomposition en sous-systèmes reste bénéfique.

Enfin, les diagrammes I.P.O. peuvent s'avérer un complément idéal du D.F.D. Ils permettent éventuellement de raffiner ce dernier en décrivant plus précisément les tâches.

Notons que les tables de décisions peuvent se révéler fort utiles lors de l'étude du système existant, notamment grâce à leur pouvoir de regroupement d'un grand nombre d'éléments.

EXCELERATOR figure, quant à lui, parmi les leaders pour l'aide qu'il apporte à la réalisation des différents

modèles. Sa rapidité d'exécution et son interface graphique le rendent agréable et efficace.

PALOMA reste un outil spécifique de la C.G.E.R. Nous ne reprendrons pas ici une critique de cet outil. Il faut tout de même admettre qu'il donne des informations très précieuses sur ce que sera la future base de données et corrige éventuellement les modèles conceptuels. De par ces aspects, il reste un bon outil de validation au terme de l'étape conceptuelle.

3.5.2. Le niveau fonctionnel

Du côté des données, le modèle logique est impératif. Sans lui, aucun schéma de base de données convenable ne pourra être conçu.

Pour établir ce modèle, un modèle dynamique peut s'avérer très utile en fournissant les accès les plus importants. De même, la normalisation permettra d'élaborer un schéma le plus adéquat possible en contrôlant la redondance.

Du côté des traitements, l'utilisation des diagrammes I.P.O. à cinq colonnes est très avantageuse pour spécifier les traitements. La spécification se fera de préférence à l'aide des tables de décision et non en pseudo-code qui est procédural, donc inadapté pour des spécifications. Quant à J.S.P., il est possible de l'utiliser pour spécifier les traitements (structure des données et des résultats et contraintes entre ces structures) sans avoir recours à un algorithme.

TELON et EXCELERATOR peuvent être utilisés tout deux à ce niveau. N'ayant pas une grande expérience de ces produits, nous n'oserons aucune classification.

3.5.3. Le niveau technique

Le formalisme JACKSON et le pseudo-code sont à peu près à égalité du point de vue efficacité de description (globale et détaillée) des traitements. Ils sont procéduraux. Ils définissent donc un algorithme dans un langage de haut niveau, multi-cibles, qu'il "suffit" de traduire dans le langages de programmation choisi.

Toutefois, on évitera d'utiliser le pseudo-code lorsqu'un grand nombre de conditions - actions sont à prendre en compte.

Les tables de décisions peuvent aussi être employées pour décrire les programmes, mais elles ne définissent pas exactement ceux-ci, en ce sens que l'ordre des actions à exécuter n'est pas toujours défini. Il se pourrait alors que, dans les cas où cet ordre a de l'importance, des erreurs risquent de se produire lors de la traduction dans le langage de programmation.

Pourtant, elles peuvent s'avérer fort utiles si les algorithmes élaborés contiennent un très grand nombre de conditions.

3.6. Conclusion

Dans ce chapitre, nous avons donné un aperçu de quelques modèles et outils d'aide au développement de logiciels. Nous avons vu qu'ils étaient souvent à la base d'une bien meilleure compréhension du réel, et qu'ils facilitaient sa représentation.

Mais, malgré toutes leurs qualités, ces modèles, ces techniques, nécessitent quelques efforts de la part de leurs utilisateurs, notamment l'adoption d'une démarche plus rigoureuse. Malheureusement, ces utilisateurs refusent de fournir des efforts supplémentaires et employent le moins de modèles et de techniques possible.

Il est dommage que ces supports de développement soient considérés de la sorte, alors que leur but est d'améliorer la qualité des logiciels.

Quoique les modèles et outils commencent à s'intégrer doucement au développement, leur efficacité maximale ne sera atteinte que lorsque les développeurs les accepteront pour ce qu'ils sont, c'est-à-dire des supports, et non des contraintes ou des assistants réfléchissant à leur place.

Chapitre 4

Intégration de S.D.M. et des acquis méthodologiques de la C.G.E.R.

4.1. Introduction

Dans le second chapitre, nous avons présenté la méthode S.D.M. d'un point de vue théorique, sans s'attacher à un environnement particulier.

Mais lorsque l'on analyse cette méthode et que l'on tente de comprendre exactement ce qui se passe dans chaque phase et dans chaque activité, certaines imprécisions apparaissent. Les frontières entre les phases sont parfois mal définies.

Ces imprécisions sont dues au fait que S.D.M. propose une "checklist" très détaillée des activités à mener lorsque l'on développe un système informatique. Malheureusement, l'importance relative des ces activités peut varier très fort en fonction des projets et des environnements. Dès lors, il n'est pas toujours aisé de différencier le primordial du secondaire.

Mais, comme nous l'avons vu au chapitre 2, c'est à l'utilisateur de la méthode de reprendre les points qui lui paraissent importants en fontion de ses problèmes.

Dans ce quatrième chapitre, nous allons tenter de restructurer cette méthode en l'intégrant à l'environnement particulier de la C.G.E.R. présenté ci dessus[32]. Cette intégration est relativement aisée vu la très grande souplesse de S.D.M. et son indépendance vis-à-vis des modèles et outils qui permet à ses adeptes de prendre uniquement ce qui leur convient tout en ignorant le reste. Nous avons donc pu enrichir les acquis de la Caisse d'Epargne et combler les lacunes grâce aux éléments de la checklist S.D.M. qui nous semblaient adéquats.

---

32. Cfr. supra chapitre 1

Nous allons également mettre en évidence les différents points de contrôle -ou points de passage- qui jalonnent le processus de développement en essayant de découvrir qui doit, ou devrait, effectuer ces contrôles.

Le but de la restructuration que nous présentons ici n'est pas de remplacer la méthode S.D.M., loin de là ! Il est simplement de fournir à ses utilisateurs néophytes un complément qui leur permettra de mieux la comprendre et de l'utiliser le plus efficacement possible. Nous insistons sur le fait qu'il faut en tout cas conserver la méthode originale à côte de cette restructuration car elle reste LA référence en cas de doute.

Signalons dès à présent que le travail exposé dans ce chapitre ne résout pas le problème de modèles et outils. Ce n'était d'ailleurs pas son objectif. Malgré la restructuration, seule une réponse à la question "Quoi faire ?" est apportée. Aucune technique nouvelle n'est apparue avec à S.D.M. La difficulté principale reste toujours de trouver une réponse à la question "Comment faire le quoi ?" pour les stades où rien n'est encore établi. Pour que la méthode et la restructuration fonctionnent parfaitement, il faudrait que tous les stades du développement soient supportés, si possible, par une technique et/ou un outils.

En ce qui concerne les étapes non encore supportées par des modèles, les développeurs devront utiliser leur expérience à bon escient et travailler avec bon sens. Ils leur sera très souvent nécessaire de se référer à la "checklist" proposée par la méthode S.D.M. [SDM] et appliquer ses recommandations plus rigoureusement.

4.2. Restructuration proposée à la C.G.E.R.

4.2.1. Présentation de la démarche

La restructuration présentée ici vise principalement à clarifier la méthode S.D.M.

En reprenant les quelques soixante-sept activités de S.D.M. [SDM], nous avons observé qu'il était possible d'en regrouper certaines et qu'il était également judicieux d'introduire un parallélisme entre elles.

Nous avons effectué les regroupements en tenant compte de deux critères.

D'une part, nous avons remarqué que certaines activités n'étaient que le prolongement ou le raffinement d'autres activités antérieures. Nous avons rassemblé ces activités au sein d'une seule nouvelle activité.

D'autre part, plusieurs activités, éparpillées dans différentes phases, traitent de problèmes similaires (par exemple les problèmes de matériel). Nous avons là aussi regroupé ces activités.

En ce qui concerne le parallélisme introduit, nous avons constaté que, premièrement, certaines nouvelles activités commençaient très tôt dans le processus de développement et se terminaient assez tard, et que, deuxièmement, d'autres activités se déroulaient souvent simultanément. Nous avons donc placé ces deux types d'activités en parallèle dans le processus de développement.

Dans le point 4.2.2 ci dessous, nous présentons les nouvelles activités obtenues par fusion d'activités de la méthode initiale S.D.M.

Pour chacune de ces nouvelles activités, nous préciserons le(s) niveau(x) d'analyse (conceptuelle, fonctionnelle, technique, ...) au(x)quel(s) elle se rattache.
Nous décrirons ensuite ce qui doit -ou devrait- se passer lors de cette activité.
Enfin, nous donnerons une liste des outils et modèles, présentés précédemment au cours du troisième chapitre, qui trouvent leur application dans la nouvelle activité ainsi définie.

Afin de faciliter au maximum la compréhension, nous proposons en annexe un schéma détaillé de type PERT décrivant l'ordonnancement des différentes nouvelles activités[33], une présentation par raffinements successifs des phases et activités de la méthode restructurée[34], ainsi qu'un tableau des références décrivant les liens entre les nouvelles activités et les activités proposées par S.D.M.[35].

4.2.2. Les nouvelles activités

4.2.2.1. Identification d'un problème

Point de départ de tout le processus de développement, l'identification d'un problème correspond à une prise de conscience qu'un disfonctionnement ou qu'un nouveau besoin existe dans l'entreprise et qu'une action serait nécessaire.

Remarquons que la pression ne vient pas toujours de chez l'utilisateur. Parfois, le changement peut découler de nécessités économiques. De plus, les

---

33. Cfr. annexe L.

34. Cfr. annexe M.

35. Cfr. annexe N.

utilisateurs ne connaissent pas toujours les possibilités des moyens informatiques.

4.2.2.2. Demande de développement

La demande de développement consiste en un recueil de renseignements généraux sur un secteur d'activités de l'entreprise concernant ses buts et objectifs, ses plannings, ses déficiences actuelles et les éventuelles solutions envisageables avec leurs avantages et leurs inconvénients. Elle est remplie par l'utilisateur de l'informatique avec éventuellement le support du C.T.I.

Le but de cette demande est d'estimer les avantages que l'on pourra retirer d'une informatisation. Dès lors, il convient de décrire clairement les avantages (quantifiables ou non) attendus d'un éventuel nouveau système par rapport à la situation actuelle.

Tout ce qui se passe lors de cette activité va servir pour la prise de décision quant au lancement du projet, et ultérieurement de base pour l'analyse conceptuelle.

Le degré de précision de la demande de développement peut varier très fort d'un projet à l'autre (une demande doit être remplie aussi bien pour un nouveau système que pour une extension mineure d'un système existant). Parfois, seul le formulaire "Demande de Développement"[36] sera rempli. Dans d'autres cas, une étude préalable aura lieu et une véritable analyse sera menée. Un cahier des charges sera déjà constitué et d'éventuels appels d'offres lancés.

Notons que l'idée d'étude préalable revient aussi dans la méthode S.D.M. qui propose une phase zéro "Etude

---

36. Cfr. annexe A.

Préalable". Il nous a semblé adéquat de reprendre la demande de développement de la Caisse d'Epargne en lieu et place de cette étude préalable dans un but de minimisation des changements qu'introduit cette nouvelle méthode.

Pour terminer, signalons que la demande de développement joue un rôle différent des autres étapes. Elle constitue simplement le déclencheur du développement.

4.2.2.3. Etude de l'existant

Cette activité va permettre d'avoir une vue d'ensemble de l'organisation, de mettre en avant les problèmes actuels et de spécifier ce que le nouveau système devra faire au minimum.

Elle se situe au niveau de l'analyse conceptuelle et considère et les données et les traitements.

On effectue ici l'analyse et l'évaluation de la situation existante. On recherche les buts et les objectifs principaux de l'entreprise -ou d'une partie de celle-ci-, on analyse les plannings prévus et les possibilités de croissance, on détermine les avantages et les inconvénients du processus de production actuel (ex : goulets d'étranglement) et enfin, on analyse le ou les éventuels systèmes d'information existants.

Comme l'étude de l'existant est à la base de la décision de continuer ou non le développement, il importe de la mener avec un certain degré de précision. Celui-ci variera en fonction de l'enjeu économique.

Notons qu'il y a toujours une étude de l'existant, et que dans les cas ou il n'y a pas d'existant, l'expérience des gens suffira pour élaborer (ou mettre à jour) un cahier des charges.

Dans beaucoup de cas, et par manque de temps, il ne sera pas possible de développer tous les points proposés ci-dessus. Il est cependant primordial d'analyser correctement la situation existante ainsi que les systèmes d'information actuels.

Lors de ces deux premières activités, qui constituent en fait les spécifications préliminaires, on identifie un problème. Comme nous l'avons fait remarquer plus haut, la pression peut venir soit de l'utilisateur, soit de l'environnement économique.

Si l'utilisateur est l'instigateur, il a peut-être déjà une ou plusieurs solutions en tête pour résoudre le problème. Certaines de celles-ci sont pure fiction, mais d'autres sont parfois beaucoup plus réalistes.

Il est essentiel de lui demander s'il pense à une solution en particulier. Dans l'affirmative, il sera alors possible de donner une première orientation au projet et de réduire ainsi le degré d'incertitude.

Le cahier des charges élaboré lors de l'étude de l'existant sera complété et terminé lors de l'analyse fonctionnelle. Son contenu peut varier d'une entreprise à l'autre et d'un projet à l'autre.

Typiquement, un cahier des charges mentionnera ce que le système doit être capable d'accomplir, c'est-à-dire ses fonctionnalités, mais aussi les ressources disponibles (financières, personnel, matériel), le temps alloué au développement et les exigences en sécurité.

4.2.2.4. Exigences et limites du nouveau système

Cette activité se déroule en phase conceptuelle et exerce une certaine influence sur les données et sur traitements.

A partir des objectifs actuels et futurs de l'entreprise, on dégage les objectifs et exigences du nouveau système d'information (performances, qualités, sécurité, 'disaster recovery') ainsi que ses limites. Il est clair que le nouveau système ne pourra pas réaliser tous les rêves. C'est donc à partir des objectifs et exigences que les limites vont pouvoir être posées (différence entre le 'must', le 'nice to have' et le 'day dream' !).

Ces limites seront posées en tenant compte de deux facteurs très importants : le temps et les ressources disponibles.
Parfois, ces deux facteurs étant insuffisants, des problèmes ne seront pas abordés.

L'"Etude de l'existant" et l'analyse des "Exigences et limites du nouveau système" sont deux activités qui se déroulent en parallèle et qui ont de fortes interactions. En effet, l'ampleur et la précision de l'étude de l'existant vont dépendre des exigences et des limites imposées au nouveau système d'information, et cette même étude de l'existant permettra de localiser certains points (certaines lacunes) qui influenceront les spécifications des exigences et limites du nouveau système[37].

A la C.G.E.R., l'étude de l'existant et l'analyse des exigences et limites du nouveau système sont réalisées par le groupe d'étude en collaboration avec le C.T.I.

---

37. Inspiré de [BOPI].

4.2.2.5. Modèle conceptuel des données

On va établir ici un modèle conceptuel des données statique, c'est-à-dire qu'on va construire un schéma Entités-Associations (ou Entités-Relations), et que pour chaque objet du schéma, on va déterminer le nombre d'occurrences.

L'introduction des volumes dans le modèle est particulier à la C.G.E.R. et est à la base de l'exécution de PALOMA[38].

4.2.2.6. Modèle conceptuel des traitements

Au cours de cette activité, on va déduire les traitements en fonction des outputs souhaités et des inputs disponibles. De plus, on va construire un modèle de structuration.

Il est intéressant de noter que la structuration des traitements permettra de localiser d'éventuels sous-systèmes qui pourront être développés indépendamment dès l'analyse fonctionnelle.

L'élaboration de ces modèles conceptuels est laissée aux soins du groupe d'étude. En cas de problèmes, ils peuvent toujours bénéficier des conseils ou de l'aide des méthodologistes du M.A.I.

Ces deux dernières activités se focalisent sur les données et sur les traitements à effectuer sur celles-ci. Elles nécessitent une collaboration intense entre informaticiens et utilisateurs afin de confronter leurs savoirs. En effet, les informaticiens ont une connaissance générale du secteur d'activités sous étude

---

38. Cfr. supra 3.4.2.

tandis que les utilisateurs excellent bien souvent dans un domaine particulier.

Les produits de ces deux activités -qui se déroulent au niveau conceptuel de l'analyse- serviront de base à l'analyse fonctionnelle et permettront d'exécuter PALOMA. Tout problème qui subsisterait devra être éliminé lors de l'analyse fonctionnelle.

Le modèle Entité-Association était une des proposition faite par la méthode S.D.M. Comme ce modèle est bien ancré à la Caisse, et donne de bons résultats, nous l'avons associé directement à la restructuration. De même, des modèles sont également en place en ce qui concerne les traitements. Ceux-ci sont bien entendu également conservés.

4.2.2.7. Elaboration de solutions-comment développer ?

Pour mettre en place le nouveau système (c'est-à-dire comment le développer[39]), plusieurs solutions sont en concurrence. Ces alternatives sont des possibilités de mise en place du système d'information avec leurs avantages et leurs inconvénients et peuvent éventuellement être accompagnées d'analyses coûts-bénéfices.

Elles tiennent généralement compte de l'environnement technique à mettre en place (télé-communication, réseaux, nouveaux appareils), des différentes manières d'effectuer le travail ou encore de l'évolution possible.

---

39. Cette étape a pour but de dire comment on va pouvoir trouver la solution, avec quels moyens. La solution en elle-même n'est trouvée qu'à partir de l'analyse conceptuelle

Par exemple, on pourrait envisager des solutions telles que :
- achat d'un logiciel tout fait;
- développement interne;
- développement en collaboration avec une firme extérieure.

Il est nécessaire de bien détailler chaque solution pour faciliter au maximum le choix. Mais il faut parfois se méfier du biais qui risquerait d'être introduit par les personnes chargées de l'activité (à la Caisse d'Epargne, le groupe d'étude), et ce, pour avantager l'une ou l'autre solution.

4.2.2.8. Choix d'une solution-qui développe ?

Pour choisir une solution parmi celles qui sont proposées, il est préférable de tenir compte d'un certain nombre de paramètres d'évaluation tels les coûts et les bénéfices, les conséquences organisationnelles et matérielles, le niveau de connaissance acquis dans le domaine des systèmes d'information, les aspects sociaux, mais aussi les problèmes de conversion si ceux-ci sont primordiaux.

D'après les renseignements que nous avons recueillis, et sans toutefois exclure les autres facteurs cités, il apparaît que, à la C.G.E.R., le choix découle d'un compromis entre, d'une part, le rapport coûts - bénéfices et, d'autre part, la garantie maximale d'obtenir un bon logiciel tout en respectant les délais.

Ces deux ultimes activités de l'analyse conceptuelle pourraient se dérouler en parallèle. En effet, il est très possible que, lors de l'élaboration de solutions, on développe un arbre avec des catégories de solutions, puis pour chaque catégorie, des sous-catégories, et ainsi de suite. Lors du développement de cet arbre,

certaines branches correspondant à des (sous-)catégories de solutions seraient élaguées car irréalisables. Une fois toutes les branches développées, on se retrouverait avec une ou plusieurs feuille(s) qui constituerait(-ent) la ou les solution(s).

Une fois la solution choisie, on établira une analyse coûts-bénéfices détaillée.

Pour faciliter la prise de décision, les personnes chargées de l'élaboration des solutions devront non seulement les détailler, mais aussi justifier pourquoi ils en préconisent une telle au détriment des autres. Cette justification éclairera les décideurs qui pourront alors choisir une solution en toute sérénité.

4.2.2.9. Exigences et limites détaillées du nouveau S.I.

Cette activité de niveau fonctionnel consiste en un raffinement de l'activité "Exigences et limites du nouveau système"[40].

Par exemple, on pourra prendre en compte des facteurs tels la durée de vie du système d'information, le personnel requis pour le faire fonctionner ainsi que d'autres facteurs, prévisibles ou non.

Cette activité peut être omise si, lors de l'analyse conceptuelle, les exigences et limites du nouveau système ont été très détaillées.

---

40. Cfr. supra 4.2.2.4

4.2.2.10. Décomposition en sous-systèmes

C'est lors de cette activité que le système va être décomposé en sous-systèmes. Cette décomposition dudit système tiendra compte de plusieurs critères tels la cohésion interne, l'interfaçage et la fonctionnalité du sous-système (tâche qui commence quelque part et qui finit quelque part).

Les sous-système pourront être développés de manière asynchrone. Pour chacun, on établira une analyse fonctionnelle et une analyse technique indépendantes. La programmation et les tests se feront également au niveau des sous-systèmes ainsi que la conversion, la mise en production et la maintenance.

Lors de cette étape, on effectuera également un test de cohérence des sous-systèmes pour vérifier que ceux-ci sont bien développées dans une même optique.

Nous insistons sur l'importance que revêt cette décomposition en sous-systèmes. Elle influencera positivement ou négativement le reste du développement et, plus tard, les conditions de maintenance. Dès lors, l'utilisation de techniques offrant des critères précis de décomposition est fortement recommandée pour parvenir à une structure optimale.

Cette technique de développement par sous-système est très utile lors de la mise en place de gros systèmes.

Prenons l'exemple d'une P.M.E. qui veut informatiser sa comptabilité et sa gestion des stocks. La décomposition fera apparaître deux sous-systèmes distincts : "Comptabilité" et "Gestion des stocks".

Prenons comme postulat que la comptabilité va être développée en premier lieu. Dès que le développement, le test d'acceptation et la conversion et mise en

production seront terminés pour ce sous-système, on pourra s'attaquer à la partie "Gestion des stocks".

Si une période relativement longue s'est écoulée entre la fin du développement du premier sous-système et le début du développement du second sous-système, il se pourra qu'une nouvelle analyse conceptuelle soit nécessaire. Dans l'alternative, on reprendra le développement de ce deuxième sous-système au niveau de l'analyse fonctionnelle.

Le test d'acceptation pour la partie "Gestion des stocks" sera plus global car il devra également vérifier la parfaite intégration des deux sous-systèmes ainsi que la cohérence des optiques choisies lors des deux développements.

4.2.2.11. Analyse des traitements

Cette analyse découle du cahier des charges.
Pour chaque tâche identifiée lors de l'analyse conceptuelle, on va déterminer les entrées et leurs origines, les sorties et leurs destinations ainsi que la règle de transformation des entrées en sorties.

C'est également ici que l'on va différencier les tâches manuelles des tâches automatiques, les tâches on-line des tâches batch.

Enfin, on décrira l'ordre d'enchaînement des traitements.

4.2.2.12. Analyse des données

On conçoit ici le modèle logique des données à partir du modèle conceptuel et des résultats fournis par l'exécution de PALOMA. Un processus de normalisation sera également appliqué, ce dernier étant indispensable en cas d'utilisation d'un SGBD relationnel.

A la Caisse, la transformation en modèle logique est le fruit du travail du D.B.A. Quant à la normalisation, elle est réalisée par le groupe d'étude, qui est le seul à pouvoir déterminer les.dépendances fonctionnelles.

4.2.2.13. Exigences en sécurité

L'aspect sécurité est très important au sein d'une organisation et il peut couvrir plusieurs domaines. A titre exemplatif, on peut citer : les différentes permissions d'accès aux données ou encore les plans de remplacement en cas de désastre.

Cette activité couvre non seulement la spécification de ces exigences (d'abord globale puis de plus en plus détaillée), mais aussi leur conception (ex : comment détecter les erreurs ?).

Par exemple, le logiciel R.A.C.F. utilisé à la C.G.E.R. permet de gérer les accès aux ressources en accordant des autorités aux utilisateurs.

Lors de cette activité, il sera donc nécessaire de déterminer qui peut accéder à quelle(s) donnée(s) et lors de quel(s) traitement(s).

L'examen des exigences en sécurité devrait débuter dès l'analyse conceptuelle et interagir plus ou moins fortement avec d'autres activités telle l'"Analyse des

traitements"[41]. Mais il est dommage que, par manque de temps, les problèmes de sécurité ne soient abordés que beaucoup plus tard au risque d'être bâclés.

4.2.2.14. Description des données

C'est ici que va s'effectuer la traduction du schéma logique élaboré lors de l'activité "Analyse des données"[42] en un schéma physique concordant au type de S.G.B.D. utilisé.

A ce stade, nous sommes au coeur de l'analyse technique, et le schéma élaboré ici décrit les adresses des données, les index, les pointeurs, etc.

4.2.2.15. Description des programmes

Du côte des traitements, on va modulariser le système. On détaillera d'abord les interfaces de ces modules, puis, pour chaque fonction de chaque module, on spécifiera les entrées, les sorties, les pré- et post-conditions ainsi que la procédure de transformation des entrées en sorties (en utilisant par exemple le langage structuré).

4.2.2.16. Interface Homme - Machine

C'est également maintenant que le design physique des écrans et des états va se dérouler, et ce, en collaboration avec les utilisateurs.

Le "quoi mettre" sur l'écran ou sur l'état a été détaillé lors de l'analyse des traitements; le "comment mettre" est abordé maintenant.

---

41. Cfr. supra 4.2.2.11

42. Cfr. supra 4.2.2.12

Ces trois dernières activités constituent la description technique complète du système. Elles ont pour but de rendre les spécifications tellement précises que les programmeurs n'auront plus qu'à les traduire dans un langage de programmation suivant des règles particulières, et cela sans avoir à poser de choix supplémentaires.

Hormis la description des données qui est, à la C.G.E.R., confiée au D.B.A., les deux autres activités sont à charge des analystes du C.T.I.

4.2.2.17. Plan de tests des programmes

Cette activité débute dès l'analyse fonctionnelle et se poursuit jusqu'en analyse technique.

On procède à l'élaboration d'un plan de tests, d'abord globalement puis en détail, et on identifie les données qui supporteront ces tests. Ces derniers seront effectués au niveau des programmes et lors de leur intégration.

Aucune documentation précise n'existe actuellement à la Caisse en ce qui concerne les tests. Notons cependant que ceux-ci doivent être élaborés scientifiquement, avec un souci de complétude. Il existe de nombreuses techniques facilitant la mise au point de tests précis et rigoureux. Il serait utile d'en choisir une (ou plusieurs) parmi elles qui s'adapterait le mieux à l'environnement en place.

4.2.2.18. Facilités requises ou imposées

Le but de cette activité est de voir quelles sont les conséquences, du point de vue matériel, de l'introduction d'un nouveau système d'information dans l'entreprise.

Dans certains cas, et ce pour diverses raisons (financières, matériel existant trop important,...), le matériel sera imposé. De ce fait, une analyse orientée dans ce sens est inutile.

Dans d'autres cas, une analyse très approfondie sera menée. En voici quelques caractéristiques.

Lors de l'analyse conceptuelle, l'étude de l'existant et l'examen des exigences et limites ont permis de dégager certains paramètres du nouveau système tels les volumes d'informations à traiter et les exigences de traitements. Ces premières estimations brutes donneront un aperçu du nouveau matériel informatique qu'il faudra acquérir.

Au cours de l'analyse fonctionnelle, on aura déjà une idée beaucoup plus précise de ce que sera le nouveau système et surtout du comment il sera réalisé. L'estimation des facilités nécessaires (nouveaux composants hardware, logiciels de base tels l'O.S. et le S.G.B.D.) sera plus nette.

Dès le début de cette analyse, le cahier des charges commencé lors de l'étude de l'existant va être complété. Ce dernier sera mis-à-jour chaque fois que de nouveaux éléments apparaîtront.

A la fin de l'activité, les facilités à acquérir devraient être complètement définies.

Notons que cette activité peut débuter dès la demande de développement et peut se poursuivre jusqu'à l'analyse technique.

4.2.2.19. Tests du matériel

Cette activité regroupe l'acquisition, l'installation et le test du nouveau matériel. Elle aura lieu d'autant plus tôt que les décisions concernant le-dit matériel seront prises rapidement, et au plus tard en parallèle avec l'analyse technique

Les tests du matériel se dérouleront en deux parties. D'abord, on vérifiera la parfaite compatibilité des composants hardware (tant entre les nouveaux composants qu'entre eux et l'ancien matériel). Ensuite, on testera les logiciels de base tels l'O.S., le S.G.B.D. et les utilitaires.

4.2.2.20. Codage

Cette activité consiste en la codification des procédures et des programmes élaborés lors de l'analyse technique.

Cette codification -qui pourrait être appelée "traduction"- n'entraîne aucune prise de décision de la par des programmeurs. En effet, ces derniers ne font que traduire des spécifications extrêmement précises en un langage compréhensible par la machine.

4.2.2.21. Tests

Pour chaque programme on compose des données de test (ces données sont normalement identifiées lors de l'activité "Plan de tests des programmes"[43]). Ensuite, on teste les programmes codés afin de vérifier leur correction en termes de logique de programmation.

Ces tests se font d'abord indépendamment au niveau de chaque programme, puis de manière incrémentale.

Ces activités de codage et de tests se déroulent bien souvent en parallèle. En effet, en cas de système modulaire, on aura généralement besoin de procédures correctes, donc testées, pour vérifier l'exactitude d'autres procédures.

4.2.2.22. Préparation du test d'acceptation

Dès les spécifications préliminaires terminées, l'utilisateur peut commencer à constituer certains cas de figure qu'il soumettra lors du test d'acceptation. Tout au long du processus de développement, il continuera à accumuler des "cas" qu'il exécutera lors du test décisif.

Il est clair que toutes les spécifications du nouveau système consignées par écrit seront des armes dans les mains de l'utilisateur pour refuser le système s'il ne respecte pas ces spécifications.

Pour éviter toute ambiguïté dans l'expression des besoins de l'utilisateur, des scénarios seront mis au point et ajoutés au contrat.

---

43. Cfr. supra 4.2.2.17

4.2.2.23. Analyse de l'impact organisationnel

Cette activité -qui peut commencer dès l'analyse conceptuelle- consiste en une description de ce que sera la nouvelle organisation autour du nouveau système d'information

Cette description couvrira plusieurs domaines.
A titre exemplatif on peut citer :

- l'identification du personnel à engager et à former
- la description des formulaires contenant des données à entrer dans le système ou destinés à recevoir de l'information venant du système;
- la conception des manuels utilisateurs;
- la conception des nouvelles procédures à créer autour du système d'information;
- l'identification de tous les problèmes humains qui découlent de l'apparition du nouveau système d'information.

Cette activité peut aussi s'occuper du matériel décentralisé. Par exemple, si une banque décide d'équiper ses agences de PC's avec logiciel "maison", il faudra tenir compte de l'installation de ces PC's. Ce matériel n'est pas pris en compte par l'activité "Test du matériel"[44] qui, elle, s'occupe du matériel nécessaire pour développer l'application.

Dans le cas de notre banque, si le logiciel est développé sur un PC, il suffit d'en acquérir un et de le tester, cela étant fait dans l'activité pré-citée.

---

44. Cfr. supra 4.2.2.19

4.2.2.24. Tests de formation

Les changements organisationnels, et principalement ceux concernant le personnel, ayant été définis, il convient de former toutes les personnes (utilisateurs, personnel de maintenance) qui graviteront en permanence autour du nouveau système.

Durant cette activité, on testera la formation telle qu'elle a été envisagée, et on observera quelles sont ses qualités et quels sont ses défauts. Ces tests s'effectueront sur un public choisi.

4.2.2.25. Test d'acceptation provisoire

Au cours de cette activité, le nouveau système va être testé à fond par des testeurs pour voir s'il répond bien à ce qu'il lui a été demandé et s'il le fait correctement.

Ce test prendra en compte tous les aspects spécifiés dans le cahier des charges tels les fonctionnalités, la sécurité et les procédures de redémarrage.

Le test d'acceptation provisoire est principalement orienté utilisateurs et est réalisé sur base de données de test.

4.2.2.26. Plan de conversion et de mise en production

Cette activité couvre tous les aspects de la conversion d'un ancien système et de la mise en production d'un nouveau système. Par exemple, on identifiera les programmes et données existants qui peuvent être récupérés moyennant quelques adaptations (formats de données différents, langage de programmation différent[45]).

Par la suite, au niveau de l'analyse fonctionnelle, on disposera de toutes les informations concernant le nouveau système. De là, on pourra raffiner le plan de conversion et de mise en production élaboré lors de l'analyse conceptuelle.

Enfin, pendant l'analyse technique, on détaillera de façon précise ce plan pour que la conversion et la mise en production se déroulent le mieux possible.

4.2.2.27. Conversion et mise en production

Cette activité consiste principalement en la conversion des données de l'ancien système et en la mise en production du nouveau système.

La conversion des données nécessite généralement une série d'opérations qui seront définies au préalable au cours de cette activité.

Au terme de la conversion et de la mise en production, le système d'information sera prêt à être exploité.

---

45. Ce choix de langage de programmation peut découler d'une politique d'entreprise qui impose un nouveau langage pour toutes les nouvelles applications. Normalement, ce choix se fait au cours de l'analyse fonctionnelle.

4.2.2.28. Test d'acceptation définitif

Le but de cette activité est de vérifier que le nouveau système s'intègre parfaitement avec les autres systèmes et s'il supporte toutes les données converties.

C'est lors de ce test que les performances vont pouvoir être mesurées et comparées avec les prévisions (y compris l'aspect recovery).

En effet, le "Test d'acceptation provisoire"[46] étant exécuté sur des données de test, il n'était pas possible de contrôler sérieusement les performances. Maintenant que le système tourne avec les données et volumes réels, les exigences telles les performances, et la sécurité peuvent être évaluées à leur juste valeur.

4.2.2.29. Formation

Le système étant parfaitement décrit et testé, la formation des personnes en relation permanente avec le système (utilisateurs, personnel technique, personnel de maintenance) peut commencer.

Cette formation se prolongera tout au long de la vie du système, vu qu'il est fort probable que celui-ci évolue au fil du temps, soit par l'ajout de nouvelles fonctionnalités, soit par une extension au sein de l'entreprise.

---

46. Cfr. supra 4.2.2.25

4.2.2.30. Gestion et maintenance du système

Dès que le nouveau système est utilisé, un suivi est indispensable pour assurer son bon fonctionnement. Ce suivi tiendra compte de plusieurs aspects tels la sécurité, la maintenance et l'évolution.

4.2.2.31. Mort du système

Tôt ou tard, le système développé ne répondra plus aux attentes. Les adaptations ne seront plus possibles. L'ancien système sera alors abandonné au profit d'un nouveau système plus performant.

4.3. Localisation des contrôles jalonnant le développement

4.3.1. Des contrôles : Pourquoi ? Par qui ?

L'importance du suivi dans l'élaboration d'un nouveau système d'information informatique est fondamentale. Ce suivi permettra notamment de maintenir le projet dans l'optique de développement initialement prévue ou encore de contrôler les retours en arrière et d'éviter parfois les abus.

Ces retours en arrière sont dus tantôt à une exécution superficielle des activités successives, tantôt à l'apparition de faits nouveaux au cours du développement du système.

Plus ils se situent en aval dans le processus de développement, plus ils coûtent cher en temps et en argent. Par exemple, si, lors de l'élaboration d'un schéma du système, l'on remarque qu'on ne dispose pas de suffisamment de détails concernant les exigences et les limites, un retour en arrière à l'activité précédente ne coûtera pas trop cher. Par contre, si l'on découvre au cours du test d'acceptation qu'il existe des anomalies

fonctionnelles, le coût d'un retour en arrière sera nettement plus élevé.

Dans cette analyse, nous expliciterons les retours en arrière en même temps que les points de contrôle[47], car c'est lors des contrôles que l'on décidera de continuer, de recommencer une ou plusieurs activités, ou encore de stopper le développement.

La réponse à la question : "Qui contrôle ?" n'est, quant à elle, pas toujours évidente.

A notre avis, le groupe d'étude a une très grande responsabilité dans le suivi du développement. Les membres de ce groupe ont généralement une bonne connaissance du problème et disposent d'une expérience qui leur permet de guider eux-même le développement et de réagir positivement en cas de problèmes. Ils devraient, par ailleurs, être capable de contrôler leur propre travail.

La hiérarchie n'est pas pour autant écartée du projet. Elle opère selon nous deux types de contrôle.

D'une part, le groupe d'étude lui fournit régulièrement des rapports sur l'état d'avancement du projet. Le contrôle effectué est plutôt passif en ce sens qu'aucune décision précise ne doit être prise. Mais il oblige les développeurs à se conformer à un planning.

D'autre part, à certaines étapes du développement, des décisions importantes doivent être prises (par exemple le choix de la solution). La hiérarchie intervient alors activement en participant au débat ou en acceptant (ou rejetant) les décisions proposées par le groupe d'étude.

---

47. Ces points de contrôle sont mentionnés sur les schéma en annexes L et M.

Nous n'avons pas la prétention de croire, ni de faire croire, que les points présentés ici constituent les seuls valables pour tout développement. Néanmoins, il nous semble qu'ils constituent une bonne base pour effectuer un suivi sérieux de l'état d'avancement d'un projet.

Pour chacun d'entre eux, un rapport particulier - synthèse des rapports résultant de l'exécution de chaque activité- doit être fourni. Il permettra de vérifier la validité du travail effectué et de prendre les décisions appropriées.

Théoriquement, un point de contrôle ne devrait engendrer un retour en arrière qu'au plus au point précédent. En effet, c'est lors de ce dernier qu'a été vérifiée la validité du travail fourni (cohérence, complétude, précision). Si ce point a été franchi, cela sous-entend que les activités qui le précèdent ont été correctement accomplies. Cependant, dans la pratique, il se peut que des éléments nouveaux apparaissent au cours du développement et nécessitent des retours beaucoup plus en amont.

Quand on développe un projet, il se peut également que, à un moment ou l'autre, l'on décide de le mettre au "frigo", c'est-à-dire de l'abandonner.

Dans l'environnement particulier de la C.G.E.R., seuls certains points de passage sont formalisés. La non continuité dans le contrôle est dû à l'absence de méthode couvrant tout le cycle de vie d'un projet. L'avènement de S.D.M. nous a permis, en collaboration avec les responsables "méthode" du M.A.I., de mieux cerner les points de passage qui seraient utiles.

Dans la présentation ci-dessous, nous préciserons, pour les points de contrôle existant, comment ils sont mis en oeuvre, alors que ceux n'étant pas encore établis ne feront l'objet d'aucune remarque particulière.

4.3.2. Présentation des points de contrôles

4.3.2.1. (P1) Validation de la demande de développement

La première chose à effectuer avant de commencer l'analyse est de vérifier qu'une demande de développement a été correctement remplie.

Dans l'environnement particulier de la C.G.E.R., elle doit être remise au C.T.I., et plus particulièrement à l'informaticien D chargé du développement, qui appréciera son importance.

Le processus de développement ne se poursuivra pas tant qu'une demande dûment complétée n'a pas été transmise aux personnes chargées de donner le feu vert (ou de dire stop ou d'exiger une autre demande).

Nous avons donc ici une première possibilité de retour en arrière ou de mise au frigo.

4.3.2.2. (P2) Evaluation de l'opportunité du projet

Après l'étude de l'existant et l'examen des exigences et des limites du système, une nouvelle évaluation va avoir lieu.

Celle-ci va permettre, d'une part, de vérifier la qualité de l'analyse effectuée et, d'autre part, de répondre à la question : "Est-ce que ça vaut la peine de continuer ?"

A ce stade, trois possibilités sont envisageables :
-soit on donne le feu vert et le développement se poursuit;
-soit on demande un approfondissement de l'analyse qui donne un retour en arrière au contrôle P1;
-soit on met le projet au frigo.

Deux types de contrôleurs peuvent intervenir ici.

Nous avons tout d'abord les membres du groupe d'étude, ou concepteurs, qui, grâce à leur expérience et à leur connaissance du problème, seront aptes à prendre la décision adéquate.

Ensuite, nous avons les responsables hiérarchiques. Ceux-ci interviendront si les concepteurs sont incapables de donner un avis motivé.

Passé P2, les retours en arrière se feront généralement au plus jusqu'en P1. En effet, il n'est pas concevable que l'on exige une reformulation de la demande de développement quand on est déjà avancé dans le projet.

Si, au cours du développement, l'on s'aperçoit que le projet est trop ambitieux ou irréalisable (budget trop limité, personnel insuffisant ou non qualifié), il est souvent possible de le réorienter ou d'opter pour le développement d'une partie seulement du projet initial.

4.3.2.3. (P3) Contrôle de la validité du schéma du S.I.

L'élaboration d'un schéma du système (modèle conceptuel des données + modèle conceptuel des traitements) étant normalement terminée, une vérification importante va avoir lieu. Grâce à elle, on va s'assurer que les exigences et limites ont été posées clairement et qu'un schéma du système à été élaboré.

Cette vérification pourra engendrer des retours en arrière plus ou moins importants (reconduite de l'activité ou respécification des exigences et limites et nouvelle étude de l'existant) ou à une mise au frigo éventuelle.

A la Caisse, deux contrôleurs interviennent.

Tout d'abord, et si son expérience est suffisante, le groupe d'étude contrôlera lui même son schéma du système.

Ensuite, le M.A.I. opérera une deuxième vérification de ce schéma avant de lancer l'exécution de PALOMA.

4.3.2.4. (P4) Cohérence de la solution

La solution sera choisie en fonction des exigences et limites posées.

Un processus itératif pourra avoir lieu si la solution choisie n'est pas raisonnable. On sera alors amené à sélectionner une autre solution. Si toutes les solutions ont été successivement sélectionnées puis rejetées, il faudra élaborer de nouvelles solutions ou au pire revoir les exigences et limites du système.

Enfin, dans un cas extrême, il se pourra que le projet soit mis au frigo.

A la C.G.E.R., ce sont les décideurs qui choisissent la solution en se basant sur les rapports fournis par le groupe d'étude. Ce dernier pourrait avantager une solution au détriment des autres. Il est donc important de vérifier l'adéquation de la solution choisie par rapport aux exigences et limites.

4.3.2.5. (P5) Terminaison de PALOMA

Ce point de contrôle se résume à la synchronisation de la terminaison de deux activités :

- -PALOMA doit avoir été exécuté;
- -une solution doit avoir été choisie.

PALOMA peut donc se terminer au plus tard ici.

Ce contrôle existe évidemment à la Caisse d'Epargne. En effet, le D.B.A. n'élaborera son modèle logique des données uniquement que s'il est en possession des résultats fournis par PALOMA.

4.3.2.6. (P6) Evaluation de l'analyse fonctionnelle

Avant d'entrer dans l'analyse technique, le travail effectué lors de l'analyse fonctionnelle doit être contrôlé et évalué. Si l'effort fourni est suffisant, c'est-à-dire si tous les éléments indispensables pour commencer l'analyse technique sont présents et valables, on continue le développement. Dans le cas contraire, on pourra recommencer une ou plusieurs activités de l'analyse fonctionnelle.

A ce stade, les fonctionnalités du système devraient être parfaitement définies et toutes les décisions concernant le "comment" devraient avoir été prises.

4.3.2.7. (P7) La programmation peut-elle commencer

Avant d'entamer l'une des phases les plus techniques du développement, à savoir la programmation, on doit disposer de toute une série de documents.

Parmi ceux-ci, on retrouvera les résultats des tests du matériel (supposé ok car P12), le plan de test des programmes complètement défini et, surtout, la description complète du nouveau système. Cette dernière devra inclure le design physique de la base de données des écrans et des états, la description des programmes/procédures suivant un formalisme défini (J.S.P., pseudo-code, langage structuré, table de décision).

Le contrôle va principalement porter sur la complétude de ces documents.

A ce niveau, un retour en arrière se limitera bien souvent à la reconduite de la phase.

4.3.2.8. (P8) Le nouveau S.I. est-il prêt à être testé

Avant d'effectuer le test d'acceptation, on doit avoir à sa disposition les résultats de tous les autres tests ainsi que les programmes codés et testés.

Deux solutions sont possibles :
-soit tout est prêt et le test peut débuter;
-soit des retours en arrière sont nécessaires. Ceux-ci se limiteront soit à de nouveaux tests des programmes, soit à la reconduite d'une ou plusieurs activités de l'analyse technique.

4.3.2.9. (P9) Le nouveau système est-il accepté

Après le test d'acceptation provisoire, les utilisateurs vont émettre un avis concernant le nouveau système et une décision va être prise. Soit le système est adopté et on peut le mettre en production, soit il n'est pas satisfaisant et des retours en arrière sont nécessaires. Plus il y aura d'activités à recommencer, plus le coût sera élevé.

Dans un cas extrême, par exemple en cas de refus d'utilisation de la part des utilisateurs, il se peut que le nouveau système soit mis au frigo.

D'après nos renseignements, ce contrôle existe à la C.G.E.R. et l'utilisateur est consulté. Toutefois, rien de précis (critères d'acceptation, de refus) n'est apparu dans la documentation que nous avons consultée.

4.3.2.10. (P10) La conversion peut-elle commencer

Avant de convertir l'ancien système et de mettre en production le nouveau système, il faut vérifier que tout a bien été planifié et préparé. Si ce n'est pas le cas, on affinera davantage le plan de conversion et de mise en production.

4.3.2.11. (P11) Contrôle de la formation

Le test de formation étant terminé, une évaluation de celle-ci va avoir lieu. Si elle est acceptable, on peut entreprendre le test d'acceptation (si les autres conditions sont présentes !). S'il existe des lacunes, on améliorera la formation et on la testera à nouveau.

4.3.2.12. (P12) Les problèmes matériels sont-ils résolus

Ce contrôle permettra de vérifier que tous les problèmes concernant le matériel sont résolus et que le travail technique (codage, test, conversion et mise en production) peut commencer.

4.3.2.13. (P13) Le système est-il acceptable

Après le test d'acceptation définitif, les performances réelles du système, c'est-à-dire sa capacité à traiter les volumes réels d'informations, ont pu être contrôlées tout comme l'aspect sécurité.

Si le système supporte ces volumes et s'il s'intègre bien avec les systèmes existants, on peut le considérer comme accepté.

Par contre, si de graves lacunes apparaissent lorsque le système tourne avec l'environnement réel, des

retours en arrière seront nécessaires. Au pire, on mettre le nouveau système au frigo.

De nouveau, nous sommes ici en présence d'un contrôle existant à la Caisse, mais ne faisant l'objet d'aucune formalisation. Un test de masse est effectivement mené, mais aucun contrôle précis n'est encore en place.

4.4. Utilité et domaines d'application de la restructuration

La restructuration ci-dessus est la solution que nous avons proposé au service M.A.I. de la C.G.E.R. afin d'amenuiser un maximum le sentiment de flou émanant de la méthode S.D.M.

Nous n'avons pas la prétention de croire qu'elle constitue à elle seule la panacée. Loin de là !

Cependant, elle permet aux développeurs d'avoir une vision tantôt globale, tantôt détaillée, des phases du développement d'un projet informatique. En cas de doute, il leur est toujours loisible de consulter la méthode initiale pour d'éventuelles confirmations.

Par ailleurs, l'introduction du parallélisme entre certaines activités met en lumière la possibilité de division du travail au sein du groupe d'étude. Sans entrer dans les détails, nous pouvons dire que celle-ci nécessiterait une coordination importante afin de synchroniser au mieux le travail des différentes équipes.

En ce qui concerne les domaines d'application de la restructuration proposée, outre le fait qu'elle a été principalement élaborée en fonction de l'environnement de la Caisse d'Epargne et outre les éventuelles différences du côté des outils et modèles de développement, il nous semble qu'elle constitue un complément idéal de S.D.M. [SDM] pour les développeurs débutant avec cette méthode.

Par contre, ce n'est, à notre avis, pas le cas pour les experts de cette méthode qui ne devraient rien découvrir de neuf dans notre travail.

Le travail de clarification que nous avons effectué atteint, à notre avis, le but recherché. Les résultats ont été accueillis favorablement par les gens du M.A.I. Malheureusement, tous les problèmes ne sont pas résolus. Un travail considérable reste à fournir en ce qui concerne les modèles à introduire en support à cette nouvelle méthode car, comme nous l'avons déjà évoqué, S.D.M. n'offre aucun standard dans ce domaine. La tâche future des méthodologistes de la C.G.E.R. sera donc de choisir parmi les nombreux modèles existants ceux qui s'adapteront le mieux à l'environnement en place.

Notons pour terminer que notre démarche ne constitue qu'un début. Seule une confrontation avec le vécu au travers d'une mise en pratique critique permettra de localiser les points forts de ce travail mais aussi ses lacunes. Peut-être nous dirigerons-nous alors vers une méthode satisfaisante et acceptée par la majorité.

PARTIE II

APPLICATION DE LA METHODE PROPOSEE A UN CAS PRATIQUE : LA GESTION DU COURRIER

Chapitre 1

Projet initial

Lors de notre passage dans le service M.A.I., nous nous sommes tout d'abord familiarisés avec l'environnement de développement de la C.G.E.R. Ensuite, nous avons entamé l'étude de la méthode S.D.M. ainsi que des modèles et outils capables de faciliter la réalisation de certaines activités. A partir de ces différents éléments, nous avons finalement proposé aux responsables du service une intégration qui pouvait, selon nous, correspondre à leurs besoins[48].

Cependant, notre étude étant principalement théorique, nous avons choisi d'appliquer la méthode proposée afin d'évaluer ses réelles capacités et, si nécessaire, de la réajuster.

Dans cette optique, nous avons consacré la deuxième partie de notre mémoire au développement d'un projet. Notre but initial était d'appliquer toutes les activités de cette méthode à un cas concret et de comparer les résultats obtenus avec ceux de l'équipe chargée de la réalisation de ce cas. Par là, nous espérions tirer quelques enseignements issus de la pratique au sujet de l'applicabilité de notre intégration (avantages et inconvénients).

L'introduction de S.D.M. étant récente à la C.G.E.R., un seul groupe de travail, ayant suivi la formation, était susceptible de nous acceuillir. C'est ainsi que nous avons été intégrés au groupe d'étude concerné par le développement d'un nouveau système dans le domaine de l'assurance-vie à la Caisse.

L'analyse conceptuelle du projet étant terminée, il nous fut proposé de comparer les résultats de cette étude avec ceux attendus si la méthode avait été utilisée. De plus on nous suggéra de participer aux interviews nécessaires à la réalisation de l'analyse fonctionnelle du projet et de collaborer à cette analyse. Remarquons qu'il nous a été impossible de trouver un projet sur lequel on puisse

---

48. Cfr. partie I.

examiner toutes les phases du développement dans les délais voulus.

Après avoir examiné le rapport de l'analyse conceptuelle et avoir participé aux premières interviews, nous avons malheureusement constaté qu'il nous serait impossible de mener à bien notre étude dans les temps impartis vu l'ampleur du projet développé (celui-ci s'étend sur plusieurs années).

Après plusieurs réunions et discussions avec notre promoteur, Monsieur LE CHARLIER, et les responsables du groupe, et vu l'impossibilité de travailler sur un projet moins volumineux, nous avons limité notre étude à l'analyse d'un sous-problème, à savoir la gestion du courrier dans le domaine de l'assurance-vie. En choisissant ce sous-problème, des perspectives de poursuivre le développement plus loin que l'analyse fonctionnelle sont apparues.

Par la suite, les renseignements sur les traitements possibles, sur les types de courriers, sur les volumes de données nous semblant insuffisants pour réaliser une analyse réellement en accord avec les désirs du groupe, nous avons choisi de développer l'application d'un point de vue plus général.

Nous avons donc abordé le problème plus globalement et nous avons réalisé l'étude d'une gestion de courriers sans aucune contrainte organisationnelle ou technique. De ce fait, le produit auquel nous sommes arrivé est, selon nous, aisément adaptable à différentes entreprises. Etant donné ce niveau de généralité, nous nous sommes limités, aux analyses conceptuelle et fonctionnelle et nous nous sommes basés sur les renseignements recueillis lors de notre participation aux interviews ainsi que sur les documents transmis par le groupe d'étude.

Signalons également que, contrairement à l'orientation initiale, nous avons utilisé la version intégrée de la

méthode[49] pour développer notre application. Mais, notre étude étant limités aux étapes de niveau fonctionnel ou conceptuel, il nous a été impossible d'aborder sérieusement certains points, notamment la préparation du test d'acceptation, les exigences en sécurité, le matériel à acquérir et l'impact organisationnel. Pour ces étapes, nous nous sommes contentés de donner quelques considérations d'ordre général.

Pour les différentes raisons mentionnées plus haut, il subsiste quelques imprécisions dans notre analyse, notamment, sur les types de courriers, sur les typologies, sur les différents paragraphes d'un courrier émis et sur les règles d'agencement de ceux-ci. Pour les activités annexes[50] aux analyses des données et des traitements, nos informations étant insuffisantes pour décrire ces activités, nous nous bornerons généralement à émettre des considérations d'ordre plus théorique.

Comme aucune implémentation n'a été effectuée, le résultat est difficilement critiquable. Cependant, malgré certaines imprécisions inhérentes au niveau d'abstraction adopté, il nous semble que le résultat est bon. Le sous-système (nous l'appelons sous-système car il est peu probable qu'il soit implémenté seul; il servira généralement de complément à un système en place ou en cours de développement) que nous avons mis au point devrait être utilisable dans des domaines de gestion relativement étendus.

Signalons pour conclure cette introduction que notre formation universitaire nous a poussé à appliquer la méthode le plus rigoureusement possible. Sans cette méthode, des étapes telles que "Analyse de l'impact organisationnel" et "Formation" n'auraient sans doute pas été abordées. Elle

---

49. Cfr. partie I, chapitre 4.

50. Ces activités annexes sont les exigences en sécurité, la préparation du test d'acceptation, l'analyse de l'impact organisationnel et le matériel à acquérir.

nous a donc guidé tout au long de notre analyse et nous pouvons avancer que, sans elle, les résultats obtenus auraient vraisemblablement été différents.

Chapitre 2

Etude d'opportunité et analyse conceptuelle

2.1. Identification d'un problème

Il est peu probable que ce soient les utilisateurs qui aient identifié les problèmes relatifs au courrier. Selon nous, une gestion automatisée du courrier découlera bien souvent de la mise en oeuvre d'un système d'information plus global.

L'idée d'une telle gestion est, à notre avis, motivée par un soucis de meilleure organisation interne de l'entreprise ayant comme conséquence directe un service de haute qualité vis-à-vis des clients.

Un autre facteur pouvant influencer ce choix est une volonté des décideurs de vouloir maintenir leur entreprise à un niveau compétitif élevé.

2.2. Demande de développement

Par soucis de respect scrupuleux de la méthodologie présentée dans la partie précédente, nous complétons ici le formulaire "Demande de Développement". Signalons simplement que son contenu pourrait varier très fort selon les environnements en place.

2. DESCRIPTION DU CONTEXTE ACTUEL.

SITUATION ACTUELLE :

Gestion de courrier classique avec classement manuel

La rédaction du courrier est tapée à la machine (ou écrite à la main)

PRODUIT(S) CONCERNE(S) :

ESTIMATION DES PRINCIPAUX VOLUMES D'INFORMATION A TRAITER

Variable suivant le volume de courriers émis et reçus gérés par le système

3. CAUSES ET ORIGINE DU PROBLEME.

Possibilités techniques attrayantes

Perte de temps pouvant résulter d'un classement manuel (recherche d'information pour les traitements de gestion)

4. AVANTAGES ATTENDUS

4.1. AVANTAGES QUANTIFIABLES.

- gain de temps lors de certains traitements (recherche, rédaction, renseignements au client, ...)

4.2. AVANTAGES NON QUANTIFIABLES.

- qualité du service au client
- satisfaction des gestionnaires
- courrier de meilleure qualité

5. CONTRAINTES DE REALISATION.

5.1. PREVOYEZ-VOUS DIFFERENTES PHASES DE DEVELOPPEMENT?

- l'analyse conceptuelle de la gestion du courrier
- l'analyse fonctionnelle de la gestion du courrier
- l'intégration à un ou plusieurs S.I. existant(s)
  1) analyse des courriers gérés par ce S.I.
  2) adaptation du système de gestion du courrier (niveau fonctionnel)
  3) réalisation
  4) tests
  5) formation & utilisation

5.2. BUDGET DE DEVELOPPEMENT ENVISAGE.

- nouveau matériel éventuel (écran A4)
- temps de développement
- nombres de personnes concernées par le développement

2.3. Etude de l'existant

Dans ce paragraphe, nous allons mener de front l'analyse et la critique de l'existant.

L'existant est une gestion de courrier classique avec des lieux de rangement déterminés, soit centralisés (tout le courrier se trouve au même endroit physique), soit décentralisés (le courrier est rangé selon son type dans les départements concernés).

Dans ce type de gestion, une limite sérieuse apparaît lorsque l'on soulève le problème du classement des courriers relatifs à un contrat. En effet, il est très difficile, voir impossible, de trier tous les courriers selon plus d'un ou deux critères. Nous verrons que dans ce domaine, une gestion automatisée offre de nets avantages.

De cette limite en découle une autre, génératrice de pertes de temps nombreuses. Le fait que peu ou pas de critères de tri soit présents nécessitera de la part des gestionnaires de relire un ou plusieurs documents avant de trouver les éléments recherchés.

Une autre perte de temps peut également être causée par les déplacements requis pour atteindre le courrier (qui se trouve, rappelons-le, à un seul endroit physique !). Si celui-ci est décentralisé, cette perte de temps peut rester minime. Par contre, si le courrier est centralisé, le temps nécessaire sera nettement plus important. Enfin, dans certaines sociétés où le courrier est vraiment vital, il se peut que certaines procédures administratives soient prévues pour consulter un courrier. Le temps perdu devient alors important.

L'unicité du courrier peut, quant à elle, être à la base de conflits d'utilisation plus ou moins nombreux suivant le type d'entreprise. Deux gestionnaires distincts ne

pourront en effet pas consulter sérieusement et en même temps le même document.

Enfin, un dernier désavantage d'une telle gestion du courrier est qu'elle n'est pas à l'abri de perte ou de destruction des documents.

A côté de ces différentes limites relatives à la gestion du courrier classique, on trouve plusieurs avantages non négligeables.

Tout d'abord, le courrier original reflète exactement la pensée du client. Aucun doute, aucune falsification ne peut exister (ou du moins difficilement).

Ensuite, l'unicité de chaque document permet de garder sa confidentialité, du moins pour les entreprises où celle-ci est nécessaire.

Enfin, le document original restera toujours le seul valable en cas de litige. Il sera l'unique référence pour authentifier les éventuelles signatures. Aucun résumé, aucune réduction de l'original ne pourront le remplacer.

2.4. Exigences et limites

Les exigences du nouveau système sont, pour la plupart, issues directement des inconvénients du système existant. Leur importance respective peut varier suivant le type d'activités exercées dans l'entreprise et supportées par le système de gestion électronique du courrier.

Une première exigence sera le multi-classement des courriers suivant une typologie définie en fonction de l'activité de l'entreprise. Cette typologie devra permettre d'accéder directement aux documents concernés et de pouvoir sélectionner les courriers selon des critères précis. Ces critères varieront en fonction des entreprises et influenceront le classement. A titre exemplatif, nous pouvons citer une sélections chronologique, par type de

courrier, par nom, par service de gestion, par contrat, etc.

Ensuite, le nouveau système devra minimiser les accès aux dossiers papier, et donc permettre un gain de temps appréciable.

Avec le nouveau système, il sera également possible à de multiples gestionnaires de consulter simultanément le même courrier. Pour atteindre ce but, il faudra prévoir un nombre suffisant de postes de travail afin d'éviter aux gestionnaires une perte de temps consécutive à l'attente de libération d'un poste.

Une exigence dont l'importance est variable est la confidentialité des documents enregistrés. Le nouveau système devra prévoir des procédures d'entrée suffisamment sérieuses pour empêcher toute consultation abusive du courrier. Dans le même ordre d'idées, la production de duplicata et/ou de copies de courriers émis fera l'objet de contrôles sérieux.

Toujours dans le domaine de la sécurité - confidentialité, le nouveau système devra offrir des possibilités poussées de récupération des données en cas de désastre.

Le nouveau système devra, suite à la réception de courriers venant des clients, permettre un déclenchement automatique de certains traitements[51]. Un système d'agenda pourra aussi être utile pour relancer les gestionnaires, ainsi qu'un utilitaire de rappel des événements non traités.

En ce qui concerne le courrier émis, l'automatisation devra accélérer le temps d'élaboration des lettres. Pour atteindre ce but, le système sera le plus convivial possible, et sa structure permettra la composition de

---

51. Par exemple expédition de renseignements demandés, mise-à-jour de la base de données, etc.

documents à partir de bibliothèques de paragraphes standards. De plus, il sera possible à tout moment de reconstruire un courrier émis (ou reçu).

Des liens étroits entre les courriers émis et les courriers reçus devront être établis afin de pouvoir reconstituer sans peine l'historique d'un contrat, d'une correspondance, ou des relations qu'a l'entreprise avec un de ses clients.

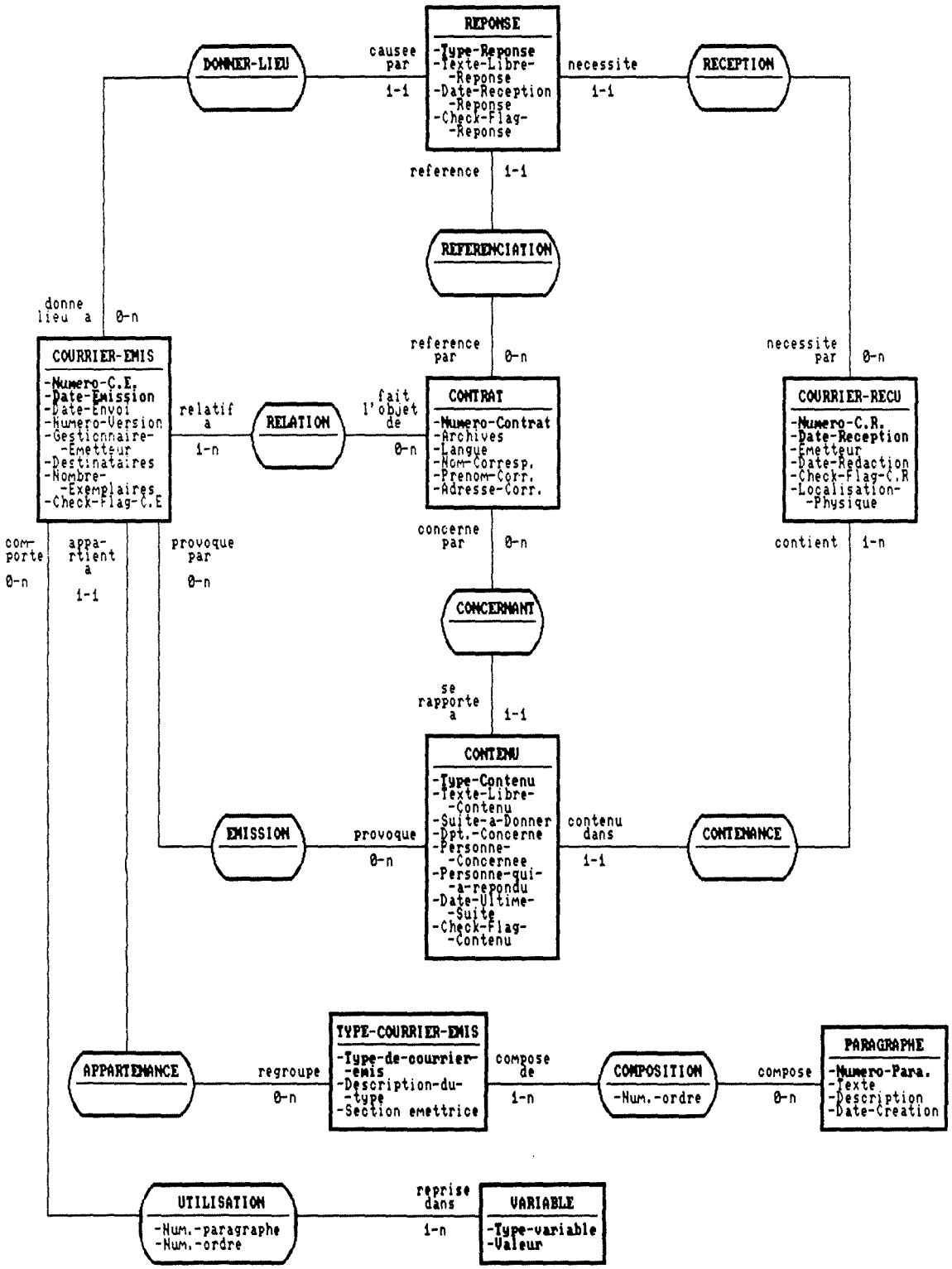
Du côté des limites du nouveau système, nous avons tout d'abord une limite typologique. Seul le courrier ayant une importance, une valeur informationnelle pour l'entreprise sera conservé.

D'autre part, le système ne sera pas capable de remplacer à 100% les dossiers papier. Il sera toujours nécessaire de garder les originaux (dans un endroit si possible protégé) car ils seront les seuls valables en cas de litige ou de problème d'authenticité.

Enfin, l'automatisation de la gestion du courrier nécessite la mémorisation d'une trace pour chaque courrier émis-reçu. La mémorisation intégrale du courrier reçu ne sera pas possible. Dès lors, les gestionnaires devront repérer dans chaque courrier les éléments importants et ignoreront les autres. Cette façon de faire risque immanquablement d'introduire un certain degré de subjectivité si des critères précis ne sont (ou ne peuvent) pas établis.

2.5. Le modèle conceptuel des données

2.5.1. Le schéma Entité - Association



Avant de poursuivre, nous tenons à mettre en évidence les hypothèses qui ont été posées pour construire ce schéma conceptuel.

- Un type de courrier émis comporte toujours les mêmes paragraphes et dans le même ordre. De ce fait, l'introduction de paragraphes libres dans une lettre serait possible, avec ce système, via les variables. On pourrait envisager un paragraphe de la bibliothèque qui soit vide et, lors de la rédaction du courrier, on le remplirait comme une variable. Par exemple, on aurait un type de courrier comme celui-ci : paragraphe vide - paragraphe défini - paragraphe vide - paragraphe défini - paragraphe vide. Ce courrier serait normalement constitué de deux paragraphes, mais ils serait toujours possible d'insérer un texte libre à n'importe quel emplacement du texte.

- Un type de courrier n'est émis que par une et une seule section émettrice.

- Une variable peut être employée dans divers courriers, plusieurs fois dans un courrier et/ou dans un paragraphe, mais elle doit être employée au minimum une fois.

Enfin, la technique utilisée pour élaborer le schéma est le modèle Entité-Association. Dans la réalité, de nombreux entretiens avec les utilisateurs auraient été nécessaire pour parvenir à mettre au point ce schéma. Pour notre part, nous nous sommes principalemnt basés sur les quelques interviews que nous avons suivies.

2.5.2. La définition des entités

a) Contrat

Un contrat établit le lien qui existe entre l'entreprise et un de ses clients. Un client peut être lié à l'entreprise par plusieurs contrats. Cette entité est mise à titre indicatif pour rattacher la gestion du courrier à l'activité de l'entreprise.

Son identifiant est : Numéro-contrat

Ses attributs sont :

- Numéro-contrat : Numéro attribué par le système à la création d'un nouveau contrat. Par exemple, le numéro pourrait être la concaténation de la date du jour et d'un numéro d'ordre dans la journée (AAMMJJxxx). Cet attribut est obligatoire.
- Nom-correspondant : Nom de la personne à qui le courrier est envoyé. Cet attribut est obligatoire.
- Prénom-correspondant : Prénom de la personne à qui le courrier est expédié.
- Adresse-correspondant : Adresse d'expédition du courrier émis. L'adresse se compose du nom de la rue, du numéro de la maison, du numéro de la boite, du code postal, de la localité et du pays. Cet attribut est obligatoire.
- Langue : Langue supportant la communication entre l'entreprise et le correspondant et éventuellement souhaitée par le correspondant. Cet attribut est obligatoire (une langue est imposée par défaut).
- Archives : Indicateur mentionnant l'existence de courriers émis et de courriers reçus archivés sur un autre support et, donc, n'étant plus repris dans le système primaire de gestion du courrier.

Remarque : Les attributs d'un contrat varient fortement suivant l'activité des entreprises. Ici, nous nous sommes contenté de reprendre ceux qui étaient directement liés à la gestion du courrier.

b) Courrier-émis

Courrier, relatif à un ou plusieurs contrats, envoyé par l'entreprise à son client, de sa propre initiative ou suite à une demande de celui-ci.

Son identifiant est : Numéro-C.E. + Date-émission

Ses attributs sont :

- Numéro-C.E. : Numéro attribué par le système à la création d'un nouveau courrier. Il est initialisé en chaque début de journée et incrémenté à chaque émission de courrier. Cet attribut est obligatoire.
- Date-émission : Date de création du courrier. Cet attribut est obligatoire.
- Date-envoi : Date d'envoi du courrier.
- Numéro-version : Numéro de la version du courrier émis. Cet attribut est obligatoire.
- Gestionnaire-émetteur : Personne responsable du contenu et de l'émission du courrier. Cet attribut est obligatoire.
- Nombre-exemplaires : Nombre d'exemplaires produits par le système. Cet attribut est obligatoire.
- Destinataire : Destinataire de chacun des exemplaires. Cet attribut est obligatoire.
- Check-flag-C.E. : indicateur signalant que toutes les demandes contenues dans le courrier émis ont été effectivement traitées.

c) Courrier-reçu

Courrier, relatif à un ou plusieurs contrats, expédié par un client à l'entreprise, de son propre chef, ou suite à une demande explicite de celle-ci. Ce courrier a de l'importance pour la gestion du ou des contrats liant les deux partenaires.

Son identifiant est : Numéro-C.R. + Date-réception

Ses attributs sont :

- Numéro-C.R. : Numéro attribué par le système à la réception d'un nouveau courrier. Il est initialisé en chaque début de journée et incrémenté à chaque émission de courrier. Cet attribut est obligatoire.
- Date-réception : Date à laquelle le courrier a été réceptionné dans l'entreprise. Cet attribut est obligatoire.
- Date-rédaction : Date figurant sur le courrier indiquant la date à laquelle le courrier a été rédigé.
- Emetteur : Nom de la personne qui a envoyé le courrier. Cette personne peut être différente de celle qui a signé le contrat. Cet attribut est obligatoire.
- Localisation-physique : Endroit physique de stockage d'un courrier reçu, c'est-à-dire l'endroit où le document papier peut être accédé. Cet attribut est obligatoire (par défaut, c'est le service de saisie).
- Check-flag-C.R. : indicateur signalant que toutes les demandes contenues dans le courrier reçu ont été effectivement traitées.

d) Réponse

Une 'Réponse' est une demande contenue dans un courrier émis qui nécessite l'envoi par le client d'un courrier.

Son identifiant est : Type-réponse
+ Identifiant ('Courrier-émis')
+ Identifiant ('Contrat')

Ses attributs sont :

- Type-réponse : Critère, dépendant de l'activité de l'entreprise, déterminant le type de document attendu. Cet attribut est obligatoire.
- Texte-libre-réponse : texte supplémentaire détaillant le type de réponse lorsque celui-ci n'est pas assez explicite.
- Date-réception-Réponse : Date à laquelle un rappel sera envoyé au client si celui-ci n'a pas encore répondu. Cet attribut est obligatoire.
- Check-flag-réponse : Indicateur signalant si le client a effectivement répondu à la demande de l'entreprise.

e) Contenu

Un contenu est une demande contenue dans un courrier reçu qui provoque un traitement par l'entreprise et/ou peut nécessiter l'émission d'un 'courrier-émis' par celle-ci.

Son identifiant est : Type-contenu
+ Identifiant ('Courrier-reçu')
+ Identifiant ('Contrat')

Ses attributs sont :

- Type-contenu : Critère déterminant le type de traitement à effectuer et/ou de courrier à émettre. Cet attribut est obligatoire.
- Texte-libre-contenu : Texte supplémentaire affinant, si besoin est, l'objet de la demande.
- Suite-à-donner : Traitement à effectuer en fonction du contenu. Cet attribut est obligatoire.
- Département-concerné : Département où sera réalisé le traitement. Cet attribut est obligatoire.
- Personne-concernée : Personne qui doit effectuer le traitement.

- Personne-qui-a-répondu : Personne qui a effectivement fait le traitement.
- Date-ultime-suite : Date ultime à laquelle l'entreprise doit donner une suite.
- Check-flag-contenu : indicateur signalant que l'entreprise a répondu à la demande du client.

f) Type-courrier-émis

Un type de courrier émis est une catégorie de courriers susceptibles d'être expédiés par l'entreprise.

Son identifiant est : Type-de-courrier-émis

Ses attributs sont :

- Type-de-courrier-émis : Cet attribut définit univoquement l'entité en donnant son type. Cet attribut est obligatoire.
- Description-du-type : Attribut décrivant clairement la catégorie de courrier repris dans.le type.
- Section-émettrice : Nom de la section habilitée à émettre ce type de courrier.

g) Paragraphe

Un occurence de l'entité représente un paragraphe susceptible d'être incorporé à un courrier émis par l'entreprise.

Son identifiant est : Numéro-paragraphe

Ses attributs sont :

- Numéro-paragraphe : Numéro identifiant un paragraphe dans la bibliothèque. Cet attribut est obligatoire.
- Texte : Texte constituant le paragraphe avec les variables. Cet attribut est obligatoire.
- Description : Description du contenu, des variables du texte.

- Date-création : Date à laquelle le texte a été introduit dans la bibliothèque.

h) Variable

Une occurence de l'entité représente une variable employée dans un paragraphe d'un courrier émis.

Son identifiant est : Type-variable + valeur

Ses attributs sont :

- Type-variable : Définition du type de la variable. Cet attribut est obligatoire.

- Valeur : Valeur prise par la variable. Cet attribut est obligatoire.

A titre exemplatif, une variable pourrait être le nom d'un client, son adresse, la dénomination d'un produit, un montant à payer, etc.

2.5.3. La définition des associations

a) Convention

R(E1,R1;E2,R2;...;En,Rn) désigne une relation définie sur les entités E1, E2, ..., En, non nécessairement distinctes, où chaque entité Ei joue le rôle Ri.

b) Référenciation (Contrat, Référencé par;
Réponse, Référence)

On a une occurrence de la relation entre 'Contrat' et 'Réponse' lorsqu'une demande contenue dans un courrier émis est enregistrée.

Connectivités :
- 1-1 du rôle 'Référence' : une demande ou une réponse contenue dans un courrier référence un et un seul contrat.
- 0-N du rôle 'Référencé par' : un contrat peut être référencé par 0, 1 ou plusieurs demandes ou réponses.

c) Relation (Contrat, Fait l'objet de;
Courrier-émis, Relatif à)

On a une occurrence de la relation entre 'Contrat' et 'Courrier-émis' lorsqu'un courrier, envoyé par l'entreprise, est enregistré.

Connectivités :
- 1-N du rôle 'Relatif à' : un courrier émis est relatif à un ou plusieurs contrats liant l'entreprise au client.
- 0-N du rôle 'Fait l'objet de' : pour un contrat, il peut exister 0, 1 ou plusieurs émission de courriers de l'entreprise.

d) Contenance (Courrier-reçu, Contient;
Contenu, Contenu dans)

On a une occurrence de la relation entre 'Courrier-reçu' et 'Contenu' pour chaque contenu identifié dans un courrier expédié par un client.

Connectivités :
- 1-N du rôle 'Contient' : un courrier reçu donné contient au moins une demande ou une réponse de la part du client.
- 1-1 du rôle 'Contenu dans' : une demande ou une réponse est contenue dans un et un seul courrier reçu.

e) Concernant (Contenu, Se rapporte à;
Contrat, Concerné par)

On a une occurrence de la relation entre 'Contenu' et 'Contrat' lorsqu'un contenu identifié dans un courrier est enregistré.

Connectivités :
- 1-1 du rôle 'Se rapporte à' : une demande ou une réponse contenue dans un courrier se rapporte à un et un seul contrat.
- 0-N du rôle 'Concerné par' : un contrat peut être concerné par 0, 1 ou plusieurs demandes ou réponses.

f) Emission (Contenu, Provoque;
Courrier-émis, Provoqué par)

On a une occurrence de la relation entre 'Contenu' et 'Courrier-émis' lorsqu'une demande ou une réponse contenue dans un courrier reçu provoque l'émission d'un courrier par l'entreprise.

Connectivités :
- 0-N du rôle 'Provoque' : un contenu particulier peut nécessiter l'émission de 0, 1 ou plusieurs courrier de l'entreprise. Pourtant, comme chaque contenu devrait normalement représenter une demande ou un fait élémentaire, n'engendrant qu'un seul traitement, la probabilité d'envoyer plusieurs courriers suite à un contenu est faible.
- 0-N du rôle 'Provoqué par' : un courrier n'est pas obligatoirement émis suite à une demande d'un client. L'entreprise peut expédier un courrier de sa propre initiative. D'autre part, un courrier peut être la synthèse de plusieurs demandes.

g) Donner-lieu (Courrier-émis, Donne lieu à;
Réponse, Causée par)

On a une occurrence de la relation entre 'Courrier-émis' et 'Réponse' pour chaque demande faite au client.

Connectivités :
- 0-N du rôle 'Donne lieu à' : un courrier ne donne pas automatiquement lieu à une demande au client, il peut s'agir d'une réponse de l'entreprise à une demande de celui-ci. D'autre part, un même courrier peut contenir plusieurs demandes.
- 1-1 du rôle 'Causée par' : une demande de l'entreprise est causée par un et un seul courrier émis.

h) Réception (Réponse, Nécessite;
Courrier-reçu, Nécessité par)

On a une occurrence de la relation entre 'Réponse' et 'Courrier-reçu' lorsque la réponse relative à une demande, contenue dans un courrier émis, est réceptionnée par l'entreprise.

Connectivités :
- 1-1 du rôle 'Nécessite' : une demande de l'entreprise nécessite obligatoirement une réponse du client.
- 0-N du rôle 'Nécessité par' : le client pouvant envoyer du courrier de sa propre initiative, la connectivité minimale est 0. D'autre part, un courrier émanant du client peut être contenir des réponses à plusieurs demandes de l'entreprise. La connectivité maximale est donc N.

i) Composition (Type-courrier-émis, Composé de;
Paragraphe, Compose)

On a une occurrence de la relation entre 'Type-courrier--émis' et 'Paragraphe' pour chaque paragraphe repris dans ce type de courrier émis.

Son attribut est :

- Numéro-d'ordre : indique quel est la place du paragraphe dans le type de courrier.

Connectivités :
- 1-N du rôle 'Composé de' : un type de courrier émis est composé d'un ou plusieurs paragraphes.
- 1-N du rôle 'Compose' : un même paragraphe peut être repris dans la composition d'un ou plusieurs types de courriers (par exemple une formule de politesse peut être commune à plusieurs types de courrier).

j) Appartenance (Courrier-émis, Appartient;
Type-courrier-émis, Regroupe)

On a une occurrence de la relation entre 'Courrier-émis' et 'Type-courrier-émis' lorsqu'un courrier émis est repris dans un type.

Connectivités :
- 1-1 du rôle 'Appartient' : un courrier émis donné n'appartient qu'à un et un seul type.
- 0-N du rôle 'Reprend' : le type reprend 0, 1 ou plusieurs courriers émis.

k) Utilisation (Courrier-émis, Comporte;
Variable, Reprise dans)

On a une occurence de l'association chaque fois qu'une variable est utilisée une fois dans un courrier émis.

Ses attributs sont :

- Numéro-paragraphe : indique dans quel paragraphe la variable est utilisée.
- Numéro-d'ordre : indique quelle est la position de la variable dans le paragraphe (utile si un paragraphe comporte plusieurs variables).

Connectivités :

- 0-n du rôle 'Comporte' : un courrier émis peu, à la limite, ne contenir aucune variable. Il peut par contre en contenir plusieurs.
- 1-n du rôle 'Reprise dans' : une variable doit être utilisée au moins une fois pour être reprise dans la mémoire du système.

2.6. Modèle conceptuel des traitements

L'élaboration du modèle conceptuel des traitements nécessite normalement l'emploi de plusieurs modèles. Cependant, vu le niveau d'abstraction important adopté pour la spécification du problème de gestion du courrier, nous n'utiliserons que le modèle I.P.O. Le S.H.A.T. n'est pas d'application car la gestion du courrier est déjà un sous-système précis. Quant au D.F.D., il est également obsolète. Un tel diagramme décrit normalement une suite d'enchaînements de traitements particuliers ainsi que les informations qu'ils véhiculent. Or dans notre cas, plusieurs traitements sont inconnus et si tentions malgré tout de construire un D.F.D., celui-ci serait tout à fait incomplet.

a) Saisie d'un courrier reçu.

Enregistrement dans la base de données de la trace d'un courrier, expédié par un client, ayant de l'importance pour la gestion. La trace sera suffisamment précise pour permettre une reconstitution du contenu du courrier papier, des déclenchements automatiques, une relance des courriers ('Contenu') non traités. Le courrier est classé dans le service de saisie.

INPUT	PROCESS	OUTPUT
-courrier papier envoyé par le client	Saisie d'un courrier reçu	-trace du courrier dans la BD

b) Traitement du contenu d'un courrier.

En fonction de la trace d'un contenu enregistrée, on effectue un traitement et/ou, le cas échéant, on émet un courrier. Suivant le type de 'Contenu', le traitement peut être déclenché automatiquement.

A titre exemplatif, le traitement pourrait être un changement d'adresse, une ouverture de dossier, une demande de renseignements, etc.

INPUT	PROCESS	OUTPUT
-trace d'un contenu -traitement à effectuer	Traitement du contenu d'un courrier	-trace modifiée du contenu -une demande d'émission

c) Création d'un courrier émis.

Enregistrement dans la base de données de la trace du courrier émis. Le courrier émis doit appartenir à un des types définis et, éventuellement, la consultation d'une base de donnée sera nécessaire pour attribuer des valeurs aux variables. La trace enregistrée doit permettre à tout moment de reconstituer le courrier émis.

INPUT	PROCESS	OUTPUT
-une demande et les valeurs des variables	Création d'un courrier émis	-trace du courrier émis dans la BD -courrier émis sous forme papier

d) Consulter les traces de courrier.

En fonction d'un filtre (type du courrier, numéro de contrat, nom du client, une date, ...), on sélectionne des traces de
courrier (reçu ou émis). On désire consulter soit un résumé du courrier,soit l'intégralité de celui-ci.

INPUT	PROCESS	OUTPUT
-un critère	Consulter les traces de courrier	-traces sélection-nées

e) Rééditer un courrier émis.

En fonction de l'identifiant d'un courrier émis, on réalise un duplicata et/ou une copie de celui-ci.

INPUT	PROCESS	OUTPUT
-identifiant d'un courrier émis	Rééditer un courrier émis	-trace du courrier émis dans la BD -duplicata/copie sous forme papier

f) Créer un paragraphe.

On introduit le texte d'un nouveau paragraphe dans la bibliothèque des paragraphes (création d'une nouvelle occurence de l'entité 'PARAGRAPHE').

INPUT	PROCESS	OUTPUT
-texte du para-graphe	Créer un paragraphe	-bibliothèque des paragraphes modifiée

g) Consultation de paragraphes.

En fonction d'un filtre (numéro de paragraphe,...), on sélectionne des paragraphes.

INPUT	PROCESS	OUTPUT
-un critère	Consultation de paragraphes	-paragraphes sélectionnés

h) Transfert de responsabilité.

Le gestionnaire initial d'un 'contenu' peut transférer la responsabilité du traitement de celui-ci. On fait l'hypothèse, ici, que lors de la saisie du courrier on a introduit le bon type de contenu et le bon département responsable (il est possible aussi que le type de contenu détermine le département concerné).

INPUT	PROCESS	OUTPUT
-identifiant d'un 'Contenu'	Transfert de responsabilité	-trace du Contenu modifiée

i) Surveillance des délais (par journée).

En fin de journée, on consulte les entités 'Contenu' et on dresse une liste de toutes celles qui ne sont pas encore traitées et qui normalement devraient être clôturées. On fait de même avec les entités 'Réponse'. Pour ces dernières, un rappel peut être expédié automatiquement au client.

INPUT	PROCESS	OUTPUT
-événement 'fin de journée'	Surveillance des délais (par journée)	-liste des traces sélectionnées -une demande d'émission d'un rappel

j) Suspendre, Annuler un courrier

En fonction d'un événement interne ou externe, on suspend ou annule un courrier. Par exemple, la réception d'une lettre de décès peut entraîner la suppression de l'envoi d'un rappel éventuel.

INPUT	PROCESS	OUTPUT
-identifiant d'un courrier reçu	Suspendre, Annuler un courrier	-liste des traces modifiées ou supprimées

k) Localiser un courrier papier.

En fonction de l'identifiant d'un courrier reçu, on visualise la localisation physique du courrier papier.

INPUT	PROCESS	OUTPUT
-identifiant d'un courrier reçu	Localiser un courrier papier	-'localisation'

l) Epuration de la base de données.

Suivant des règles de gestion (délai de conservation des documents, règlementation, loi, etc.), on supprime la trace de courriers (reçus ou émis) de la base de données. Ces
traces sont transférées sur un autre support d'accès non direct.

INPUT	PROCESS	OUTPUT
-règles de gestion	Epuration de la base de données	-BD épurée -traces sélection-nées archivées sur un autre support

m) Dire si un courrier est totalement traité.

En fonction de l'identifiant d'un courrier (reçu ou émis), on indique s'il est totalement traité ou si celui-ci est en cours de traitement.

INPUT	PROCESS	OUTPUT
-identifiant d'un courrier	Dire si un courrier est totalement traité	-'traité OUI/NON'

2.7. Identification des possibilités de développement

Nous allons maintenant tenter de donner quelques possibilités de développement de l'application que nous étudions. Pour chacune d'entre elles, nous donnerons notre avis en tenant compte de plusieurs critères, à savoir, l'adéquation du logiciel, son coût et le temps de développement estimé.

Pour élaborer de telles solutions, le groupe d'étude devrait effectuer des enquêtes auprès de maisons de logiciels afin de rassembler toutes les informations qui leur sont nécessaires.

Une première solution consisterait à rechercher sur le marché un logiciel répondant à toutes les exigences identifiées. Cette possibilité serait quasi immédiate. Une adaptation serait sans doute nécessaire, mais si le produit choisi est intégrable, celle-ci devrait prendre nettement moins de temps qu'un développement complet.

Par contre, l'adéquation du produit ne sera pratiquement jamais parfaite. Il est vraissemblable que les fonctionnalités du logiciel choisi ne seront pas exactement celles souhaitées.

Le coût du logiciel variera selon sa qualité et ses fonctionnalités. Nous n'en connaissons aucun. Il nous

est dès lors impossible de donner de chiffres plus ou moins précis. A notre avis, plusieurs dizaines de milliers de francs seront nécessaires à l'acquisition d'un tel logiciel.

Une deuxième solution serait de confier le développement du gestionnaire de courrier au centre informatique de l'entreprise (si elle en possède un). L'avantage incontestable de cette solution (si le développement est bien mené) sera l'adéquation parfaite du logiciel aux besoins.

Malheureusement, le coût d'une telle approche risquera d'être élevé, surtout si les développeurs n'ont que très peu d'expérience. L'absence de méthodologie de développement de logiciels influencera également négativement le coût.

Quant au temps nécessaire pour mettre au point ce nouveau produit, il sera aussi fonction de la qualité des développeurs et de l'environnement dans lequel ils évolueront.

Une dernière approche concevable est de confier l'élaboration du logiciel à une société extérieure. Les connaissances qu'apportera cette société pourront se révéler précieuses pour l'entreprise.

L'adéquation du logiciel sera normalement très bonne. La société extérieure tiendra compte de tous les desiderata, et le produit sera aussi "sur mesure".

Avec une telle solution, un budget important doit être envisageable. Le prix demandé par de pareilles sociétés sont, en effet, presque toujours très élevés.

Le temps de développement nécessaire sera lui aussi variable suivant la société choisie et son expérience dans le domaine concerné. Il pourra s'avérer très concurentiel, ou être tout à fait inacceptable.

2.8. Choix d'une solution (qui va développer ?)

Le choix de la solution est normalement le fait de la direction. Quant à nous, nous pouvons tout au plus émettre un avis concernant les diverses solutions. L'environnement de développement étant indéfini (qualité du centre informatique inconnue, budgets inconnus, méthodes et standards de développement inconnus), cet avis reste vague.

Si le centre informatique est de qualité, et si les développeurs ont le temps, le développement interne nous semble offrir le meilleur compromis coût-adéquation-temps. Le logiciel élaboré sera également plus facilement intégrable aux systèmes existants.

L'achat d'un logiciel tout fait n'est jamais une bonne solution quand le problème est trop spécifique. La gestion du courrier est par excellence un problème nouveau et très particulier. Le choix d'un logiciel (s'il en existe) doit dès lors être limité, et son coût excessif.

Quant au développement en collaboration avec une société extérieure, les considérations restent du même ordre. Si cette société a de l'expérience et qu'elle propose de réaliser un produit adéquat dans les délais et à un prix raisonnable, la solution est envisageable.

Ces considérations, quoiqu'imprécises et générales, guideront normalement les décideurs dans leur choix.

Chapitre 3

ANALYSE FONCTIONNELLE

3.1. Exigences et limites détaillées

3.1.1. Objectifs vis-à-vis de la clientèle

Un des buts de l'introduction d'un gestionnaire de courrier électronique est d'améliorer les services rendus aux clients. Cette amélioration globale va se traduire par un ensemble d'objectifs particuliers que nous énonçons ci-dessous.

- Le premier but du nouveau système sera d'assurer un service de courrier rapide à destination du client. Un minimum de retard sera souhaité. Le système sera donc apte à supporter des charges qui pourront atteindre une certaine importance selon les périodes (dépend de l'activité).

- Grâce à l'automatisation de la recherche et grâce au multi-classement (par numéro de contrat, par nom, par type de demande, etc.), les gestionnaires seront capables de fournir des renseignements directement par téléphone aux clients. Cela signifie que le système devra travailler en temps réel. Un temps de réponse de l'ordre de quelques secondes à deux ou trois dizaines de secondes en temps de charge devrait être acceptable.

- Pour que le courrier envoyé aux clients reste clair et précis, il est nécessaire d'étudier avec soin les paragraphes qui constitueront les bibliothèques. Les règles d'assemblage de ceux-ci devront éviter toute contradiction, toute redondance et toute ambiguïté. Il sera également possible aux gestionnaires d'incorporer des données particulières dans certains paragraphes ou d'en construire un ou plusieurs via leur terminal. De plus, pour les entreprises exerçant une activité dans tout le pays, le système devra rédiger les documents dans la langue choisie par le client.

La technique d'assemblage de paragraphes devra, en outre, assurer une uniformité de style pour tous les documents.

Une confidentialité maximum devra être garantie pour éviter toute fuite d'informations personnelles concernant le client. Pour ce faire, l'accès interne au système sera strictement réglementé et la possibilité d'établir une "barrière" empêchant toute intrusion externe devra être étudiée soigneusement.

3.1.2. Objectifs vis-à-vis des services de gestion

Le système aura comme but principal la minimisation des consultations des dossiers papier. Pour atteindre ce but, il devra mémoriser un certain nombre de données (un minimum de données avec une signification maximum) ayant de la signification pour l'entreprise. Cette mémorisation suivant une typologie (par service de gestion, par date, etc.) assurera un suivi aisé des dossiers.

La gestion automatisée du courrier ne devra en aucun cas limiter ou réduire les fonctions de gestion courante existantes dans l'entreprise. Toutes les fonctions nécessaires devront être assumées par le système.

Comme le nouveau système viendra en remplacement de la gestion traditionnelle, il est utile qu'il offre des facilités d'apprentissage (et notamment une aide "on-line") et une bonne convivialité, et ce, pour éviter toute résistance de la part des futurs gestionnaires utilisateurs.

Toujours dans le même ordre d'idées, le système devra offrir des possibilités de recherche poussées suivant une typologie, ou chronologique. Il permettra aussi de reconstituer l'historique d'un contrat (retrouver, dans l'ordre, tous les courriers émis ou reçus relatifs à un

contrat ou à un client). Toutes ces recherches devront s'effectuer dans un laps de temps très court, notamment par la suppression des déplacements nécessaires pour accéder aux dossiers papier et de la recherche de l'information dans ce dossier, permettant ainsi aux gestionnaires de réaliser un nombre élevé de requêtes en un minimum de temps.

L'accès à un même dossier via le nouveau système par de multiples gestionnaires sera permis.

En ce qui concerne l'utilisation du système, il faudra prévoir un nombre suffisant de "stations de travail" pour permettre une introduction (ou consultation ou production) de données (ou courriers, ou traces) décentralisée et sans perte de temps (une station par gestionnaire ?).

Enfin, un dernier objectif vis-à-vis des services de gestion est l'introduction d'une standardisation des documents émis. Celle-ci permettra, outre tous les avantages de gestion pure, de donner une image de sérieux et de continuité à l'entreprise qui ne pourra que satisfaire les clients.

3.1.3. Objectifs vis-à-vis de l'informatique

Le nouveau système devra être conçu de telle sorte qu'il autorisera des modifications (ou des ajouts) fonctionnelles en n'entraînant qu'un minimum de modification du programme. Une paramétrisation poussée des procédures qui composeront le système sera donc souhaitée.

La standardisation des supports des courriers devra être maximale afin de réduire le travail des services opérationnels.

Enfin, le système devra tourner si possible sur un matériel déjà existant, et ce, afin d'éviter tout investissement inutile.

3.1.4. Objectif vis-à-vis de la direction

Le nouveau système devra être capable de fournir rapidement les données que peut requérir la hiérarchie. Entre autre, des renseignements tels la productivité de chacun des services, le temps de traitement d'un dossier et le temps de réponse devront être extraits du système sur demande.

3.1.5. Contraintes générales

Les développeurs devront tenir compte de la législation, surtout pour ce qui est de l'authenticité des documents, la production de copies, de duplicata, la légalité même de gérer le courrier par ordinateur.

Le passage d'une gestion manuelle à une gestion automatique devra être transparent au client. Celui-ci remarquera éventuellement des améliorations, mais il ne pourra en aucun cas émettre des plaintes du fait du nouveau système.

Pour éviter toute perte totale suite à un désastre, des back-up réguliers seront imposés. Leur fréquence variera suivant le taux de modification et/ou d'ajout de courrier dans le système. Si les courriers sont vraiment essentiels pour l'entreprise, il sera peut-être impératif de prévoir un double système de mémorisation pour réduire au maximum les chances de perte.

Le système devra être fourni avec une documentation complète (conceptuelle, fonctionnelle, technique et utilisateur) afin de le rendre compréhensible et exploitable par ses utilisateurs.

3.1.6. Limites du nouveau système

Les limites restent celles énoncées au cours de l'analyse conceptuelle.

3.2. Découpe en sous-systèmes

Comme nous l'avons déjà signalé lors de l'analyse conceptuelle des données, le système de gestion de courrier automatisé constitue à lui seul un sous-système. Une découpe plus fine nous semble dès lors superflue.

3.3. Analyse des données

3.3.1. La normalisation

Nous allons à présent vérifier le degré de normalisation du schéma conceptuel des données en utilisant la technique proposée dans l'intégration. Pour chaque entité du schéma, nous allons successivement revoir les critères des trois premières formes normales. Nous ne tiendrons pas compte des quatrième et cinquième formes, celles-ci étant trop spécifiques pour que nous les prenions en considération ici. Si des anomalies apparaissent, nous les corrigeront. Les relations résultantes devraient alors constituer une bonne base pour l'élaboration d'un schéma logique des données de type M.A.G.

Remarque : les identifiants des relations sont soulignés.

1) **CONTRAT** (numéro-contrat, archives, langue, nom-correspondant, prénom-correspondant, adresse-correspondant)

Cette relation est en 1FN car elle ne comporte aucun attribut répétitif ni d'attribut-relation.

Comme la 2FN ne touche que les relations à clé composée, elle ne s'applique pas ici.

Par contre, la relation n'est pas en 3FN, car l'attribut "adresse-correspondant" ne dépend pas totalement de l'identifiant. Elle dépend aussi de "nom-correspondant". Cependant, afin de ne pas outrepasser les frontières de notre système en introduisant la notion d'un fichier signalétique client, nous conserverons cette anomalie de redondance qui, à note avis, n'est pas catastrophique.

Signalons toutefois que, lors de l'intégration de ce sous-système à d'autres systèmes, il sera éventuellement nécessaire de tenir compte d'un fichier client existant.

2) **COURRIER-EMIS** (numéro-courrier-émis,
date-émission, date-envoi,
numéro-version, gestionnaire-émetteur,
destinataire, nombre-exemplaires,
check-flag-courrier-émis)

La relation est en 1FN, en 2FN et en 3FN.

3) **COURRIER-RECU** (numéro-courrier-reçu,
date-réception, émetteur,
date-rédaction,
check-flag-courrier-reçu,
localisation-physique)

Cette relation est parfaitement normalisée.

4) **REPONSE** (numéro-courrier-émis, date-émission,
numéro-contrat, type-réponse,
texte-libre-réponse, date-réception-
-réponse, check-flag-réponse)

Cette relation est également en 3FN.

5) **CONTENU** (<u>numéro-courrier-reçu, date-réception, numéro-contrat, type-contenu</u>, texte-libre-contenu, suite-à-donner, département-concerné, personne-concernée, personne-qui-a-répondu, date-ultime-suite, check-flag-contenu)

Une dépendance pourrait exister de "département-concerné" vers "personne-concernée" et "personne-qui- -a-répondu". Cependant, l'éclatement de la relation ne parraissant pas bénéfique nous la maintiendrons dans cet état.

6) **TYPE-COURRIER-EMIS** (<u>type-de-courrier-émis</u>, description-du-type, section-émettrice)

Cette relation est normalisée.

7) **PARAGRAPHE** (<u>numéro-paragraphe</u>, texte, description, date-création)

est une bonne relation.

8) **VARIABLE** (<u>type-variable, valeur</u>)

est également une bonne relation.

Comme les associations 'UTILISATION' et 'COMPOSITION' sont porteuse, il convient d'en faire des relations.

9) **UTILISATION** (<u>numéro-courrier-émis, date-émission, type-variable, valeur, numéro-paragraphe numéro-d'ordre</u>)

qui est déjà normalisée.

10) **COMPOSITION** (type-de-courrier-émis,
numéro-paragraphe, numéro-d'ordre)

est également normalisée.

3.3.2. Le modèle logique des données

Le modèle logique que nous présentons ci-dessous est le M.A.G. Ce modèle n'est pas ciblé, c'est-à-dire qu'il est normalement adaptable à tous les S.G.B.D.

Pour transformer ce schéma en un schéma conforme à un S.G.B.D. particulier, il convient de connaître les contraintes qui lui sont inhérentes. Par exemple, si le S.G.B.D. choisi n'autorise pas les types de chemins N-N, il sera nécessaire de modifier le schéma ci-dessous.

3.3.2.1. Conventions

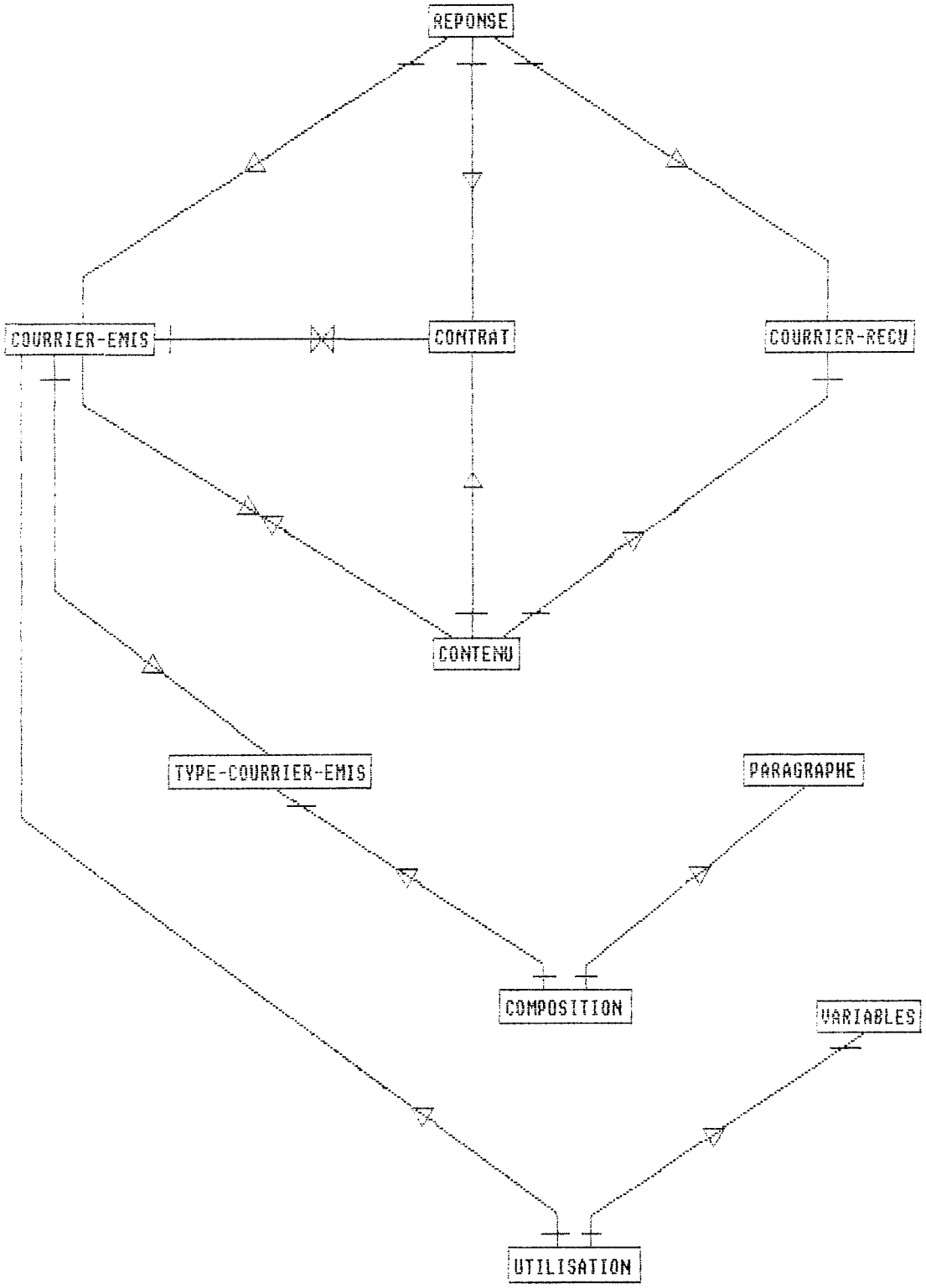
Un type d'articles est représenté par une boîte dans laquelle figure le nom du type.

Un type de chemins est représenté par un arc entre les types d'articles concernés.
Si la classe fonctionnelle de l'arc est 1-N ou N-1, on place un triangle sur l'arc, la pointe orientée vers le type d'articles "1";
Si la classe fonctionnelle est N-N, on place un diabolo (double triangle) sur l'arc.

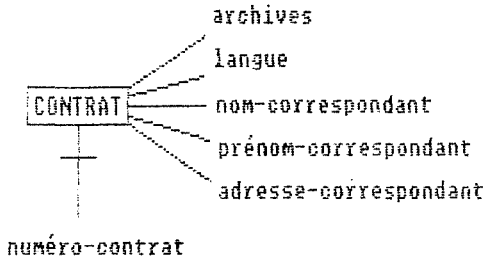
Un item est représenté par son nom et est relié au type d'article par un arc.
Si l'item est obligatoire, une barre verticale est placée sur l'arc, près de la boîte;
Si l'item est répétitif non-identifiant, on place un diabolo sur l'arc.
Si l'item est simple non-identifiant, on place un triangle sur l'arc.

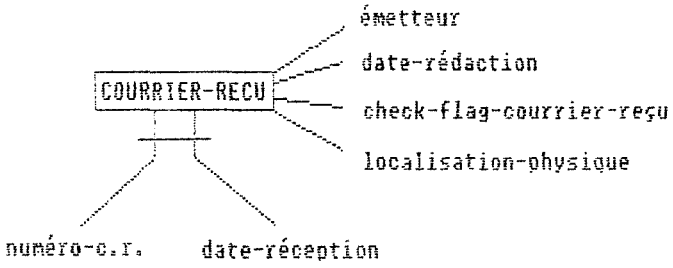
Si l'item est identifiant simple, on ne place aucun symbole.
Si l'item est identifiant répétitif (exemple : un numéro de téléphone; plusieurs numéros peuvent référencer une même personne), on place un triangle sur l'arc, la pointe tournée vers l'item;
Pour les identifiants composés, on introduit un parallélisme partiel entre les arcs.

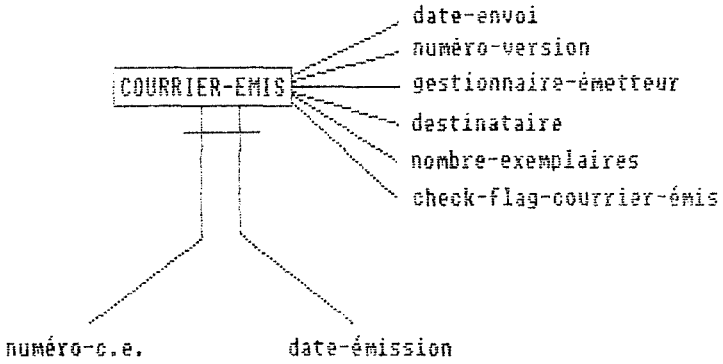
3.3.2.2. Le schéma M.A.G.

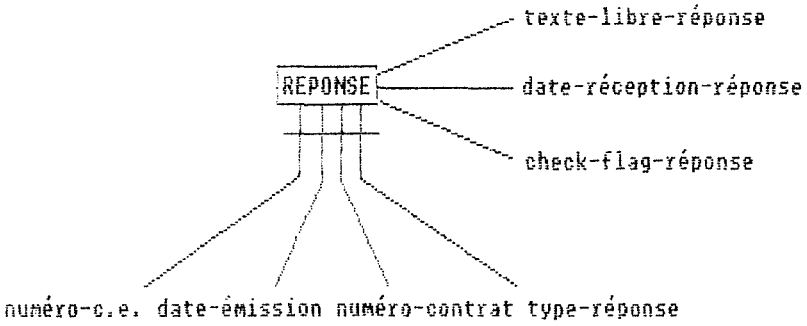


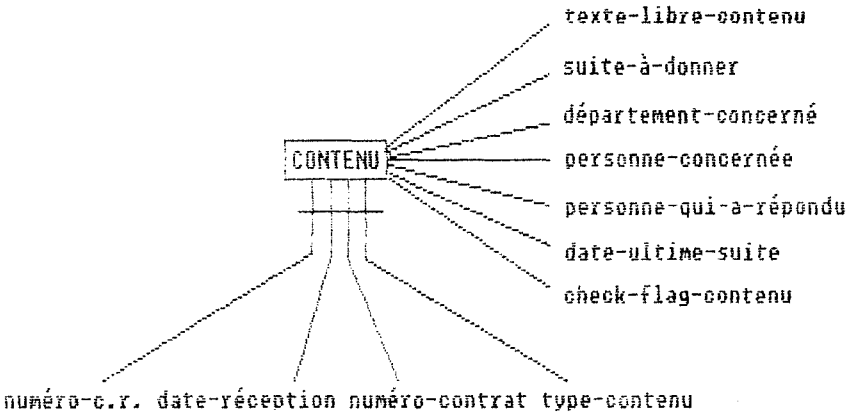
3.3.2.3. Les types d'articles du M.A.G.

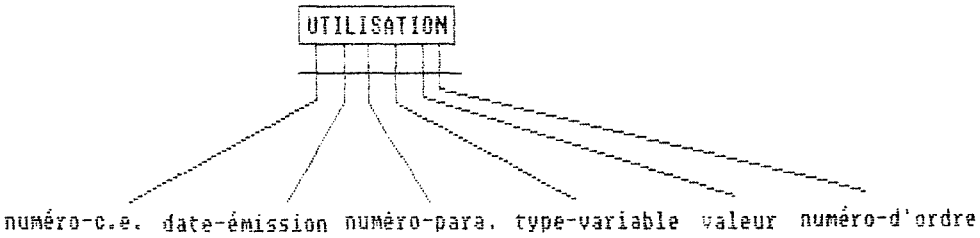


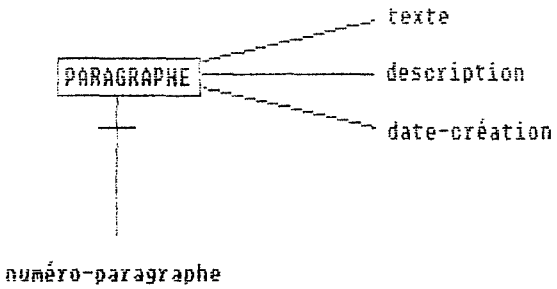


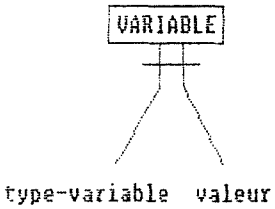


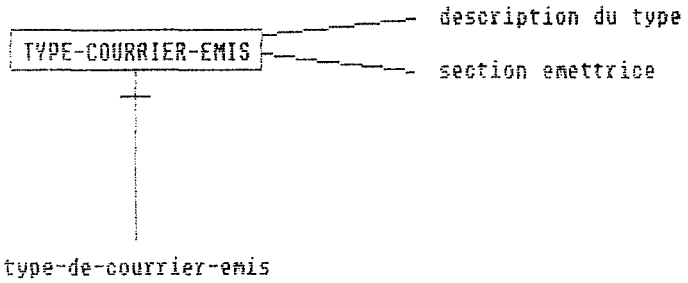


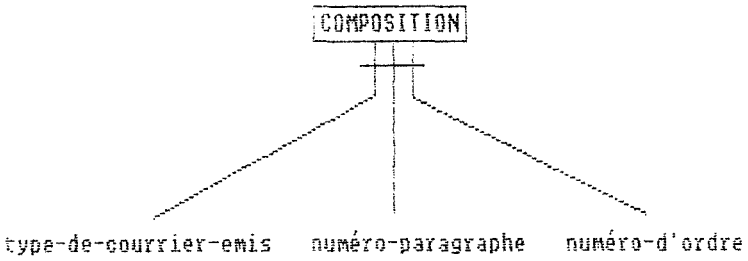

texte-libre-contenu
suite-à-donner
département-concerné
CONTENU
personne-concernée
personne-qui-a-répondu
date-ultime-suite
check-flag-contenu
numéro-c.r.
date-réception
numéro-contrat
type-contenu


UTILISATION
numéro-c.e.
date-émission
numéro-para.
type-variable
valeur
numéro-d'ordre


texte
PARAGRAPHE
description
date-création
numéro-paragraphe


VARIABLE
type-variable
valeur


TYPE-COURRIER-EMIS
description du type
section emettrice
type-de-courrier-emis


COMPOSITION
type-de-courrier-emis
numéro-paragraphe
numéro-d'ordre

3.3.2.4. Les contraintes d'intégrité

numéro-contrat (REPONSE) IN numéro-contrat (CONTRAT)

numéro-contrat (CONTENU) IN numéro-contrat (CONTRAT)

numéro-courrier-émis, date-émission (REPONSE) IN
numéro-courrier-émis, date-émission (COURRIER-EMIS)

numéro-courrier-reçu, date-réception (CONTENU) =
numéro-courrier-reçu, date-réception (COURRIER-RECU)

numéro-paragraphe (COMPOSITION) in
numéro-paragraphe (PARAGRAPHE)

type-de-courrier-émis (TYPE-COURRIER-EMIS) =
type-de-courrier-émis (COMPOSITION)

numéro-courrier-émis, date-émission (COURRIER-EMIS) =
numéro-courrier-émis, date-émission (UTILISATION)

type-variable, valeur (UTILISATION) =
type-variable, valeur (VARIABLE)

3.4. Analyse des traitements

Nous donnons ici une spécification de tous les traitements identifiés lors de l'analyse conceptuelle. Pour chaque traitement, nous donnons ses objectifs, le ou les messages d'entrées -c'est-à-dire les éléments nécessaires à son exécution-, les messages de sorties possibles -c'est-à-dire les éventuelles informations communiquées au gestionnaire-, ainsi que les modifications apportées à la base de données. Ces modifications pourront parfois varier en fonction des exigences propres à une organisation.

a) Saisie d'un courrier reçu.

Objectifs :

Enregistrement dans la base de données de la trace d'un courrier, expédié par un client, ayant de l'importance pour la gestion. La trace sera suffisamment précise pour permettre une reconstitution du contenu du courrier papier, des déclenchements automatiques, une relance des courriers ('Contenu') non traités. Le courrier est classé dans le service de saisie.

Messages d'entrée :

Un courrier papier contenant :
- Numéro-contrat
- Date-rédaction
- Emetteur
- un ou plusieurs Type-contenu
- un ou plusieurs Texte-libre-contenu

Messages de sortie :

sortie : Numéro-C.R. + Date-réception

Effets sur le S.I. :

La saisie du courrier provoque :
- la création d'une entité 'Courrier-reçu',
- la création d'une ou plusieurs entités 'Contenu',
- la création d'une ou plusieurs occurrences des associations 'Contenance' et 'Concernant'.

Si ce courrier est une réponse à une ou plusieurs demandes de l'entreprise (dans un pareil cas, lors de la saisie des contenus d'un courrier reçu, on rapproche la typologie des contenus avec celle des réponses attendues. Si un contenu est une réponse à une demande de l'entreprise, on crée le lien automatiquement si possible. Par exemple, on pourra lier la réception d'un paiement avec la demande de celui-

ci. Les rapprochements sont issus de règles de gestion.), la saisie provoque également :
- la création d'une ou plusieurs occurrences de l'association 'Réception',
- la modification d'une ou plusieurs entités 'Réponse' (modification du Check-flag-réponse),
- la modification éventuelle d'une ou plusieurs entités 'Courrier-émis' si l'on vient de recevoir la dernière réponse à des demandes contenues dans un même courrier émis (modification du Chek-flag-C.E.).

b) Traitement du contenu d'un courrier.

Objectifs :

En fonction de la trace d'un contenu enregistrée, on effectue un traitement et/ou, le cas échéant, on émet un courrier. Suivant le type de 'Contenu', le traitement peut être déclenché automatiquement.

Messages d'entrée :

Une trace de 'Contenu' contenant :
- Numéro-C.R.
- Date-réception
- Numéro-contrat
- Type-contenu
- Texte-libre-contenu (s'il existe)
- Suite-à-donner
- Département-concerné
- Personne-concernée (si elle est identifiée)
- Date-ultime-suite (si non fixée par le système)

Messages de sortie possibles :

sortie 1 : "le contenu est traité"

sortie 2 : "le contenu est en attente"

sortie 3 : Une demande d'émission de courrier

Effets sur le S.I. :

Le traitement du contenu provoque :
- la modification d'une entité 'Contenu' (modification des attributs : Chek-flag-contenu, personne-qui-a-répondu).

Si l'on vient de traiter le dernier contenu d'un courrier reçu, le traitement du contenu provoque également :
- la modification d'une entité 'Courrier-reçu' (modification du Check-flag-C.R.).

c) Création d'un courrier émis.

Objectifs :

Enregistrement dans la base de données de la trace du courrier émis. Ce courrier doit appartenir à un type défini. De plus, la consultation d'une base de donnée pour attribuer des valeurs aux variables sera éventuellement nécessaire. La trace enregistrée doit permettre à tout moment de reconstituer le courrier émis.

Messages d'entrée :

Une demande d'émission d'un courrier contenant :
- Numéro-contrat
- Langue
- Type-document
- Numéro-version
- Gestionnaire-émetteur
- Nombre-exemplaires
- Destinataire

Les valeurs des variables concernées.

Messages de sortie possibles :

sortie 1 : courrier émis sous forme papier

sortie 2 : Numéro-C.E. + Date-émission

Effets sur le S.I. :

L'émission du courrier provoque :
- la création d'une entité 'Courrier-émis',
- la création d'une ou plusieurs entités 'Variable'

- la création d'une ou plusieurs occurrences des associations 'Relation', 'Utilisation' et, 'Appartenance',
- la création éventuelle d'une ou plusieurs entités 'Bibliothèque-Paragraphes' si l'on introduit un texte libre dans le courrier.

Si le courrier émis nécessite une ou plusieurs réponses du client, cela suscite également :
- la création d'une ou plusieurs entités 'Réponse',
- la création des associations 'Donner-lieu' et 'Référenciation' relatives à 'Réponse'.

Si la demande d'émission est relative au traitement d'un ou plusieurs 'Contenu', cela implique aussi :
- la création d'une ou plusieurs occurrences de l'association 'Emission'.

d) Consulter les traces de courrier.

Objectifs :

En fonction d'un filtre (type de courrier, numéro de contrat, nom du client, une date, ...), on sélectionne des traces de courrier (reçu ou émis). On désire consulter soit un résumé du courrier,soit l'intégralité de celui-ci.

Messages d'entrée :

Un critère

Messages de sortie :

sortie 1 : Numéro-C.E. + Date-émission + Date-envoi-C.E. + Type-document + Description + Numéro-version + Section-émettrice + Gestionnaire-émetteur + Nombre-exemplaires + Destinataire + Check-flag-C.E.

sortie 2 : Numéro-C.R. + Date-réception + Date-rédaction + Emetteur + Localisation-physique + Check-flag-C.R.

sortie 3 : sortie 1 + Numéro-contrat + Texte + variables + Type-réponse + Texte-libre-réponse + Check-flag-réponse + Type-contenu + Date-rédaction

sortie 4 : sortie 2 + Numéro-contrat + Type-contenu + Texte-libre-contenu + check-flag-contenu + Type-réponse

Effets sur le S.I. :

La consultation d'une trace provoque :

1) pour un courrier reçu :
- la consultation d'une entité 'Courrier- reçu'.

Si l'on désire une trace détaillée du courrier, cela implique aussi :
- la consultation d'une ou plusieurs occurrences des entités 'Contrat', 'Contenu', 'Réponse', 'Courrier-émis', 'Paragraphes', 'Variables'.

2) pour un courrier émis :
- la consultation des entités 'Courrier-émis', 'Variables' et 'Type-courrier-émis' et des associations 'Utilisation' et 'Composition'.

Si l'on désire une trace détaillée du courrier, cela suscite également :
- la consultation d'une ou plusieurs occurrences des entités 'Paragraphes', 'Bibliothèque-Paragraphes', 'Contrat', 'Réponse', 'Contenu' et 'Courrier-Reçu',

e) Rééditer un courrier émis.

Objectifs :

En fonction de l'identifiant d'un courrier émis, on réalise un duplicata et/ou une copie de celui-ci.

Messages d'entrée :

- Numéro-C.E.
- Date-émission

Messages de sortie :

sortie 1 : duplicata/copie sous forme papier

sortie 2 : Numéro-C.E. + Date-émission

Effets sur le S.I. :

La réédition d'un courrier émis provoque :
- la consultation d'une entité 'Courrier-émis',
- la consultation d'une ou plusieurs entités 'Paragraphes' et 'Type-courrier-émis',
- la création d'une entité 'Courrier-émis',
- la création d'une ou plusieurs occurrences des associations 'Relation', 'Utilisation' et 'Appartenance'.

f) Créer un paragraphe.

Objectifs :

On introduit le texte d'un nouveau paragraphe dans la bibliothèque des paragraphes.

Messages d'entrée :

- Texte
- Description

Messages de sortie :

sortie 1 : description du paragraphe sous forme papier

sortie 2 : Numéro-paragraphe

Effets sur le S.I. :

La création d'un paragraphe provoque :
- la création d'une entité 'Paragraphe'

g) Consultation de paragraphes.

Objectifs :

En fonction d'un filtre (numéro de paragraphe, date, ...), on sélectionne des paragraphes de la bibliothèque.

Messages d'entrée :

Un critère

Messages de sortie :

sortie : Numéro-paragraphe + Texte + Description + Date-création

Effets sur le S.I. :

La consultation de paragraphes provoque :
- la consultation d'une ou plusieurs entités 'Paragraphes'

h) Transfert de responsabilité.

Objectifs :

Le gestionnaire initial d'un 'contenu' peut transférer la responsabilité du traitement de celui-ci. On fait l'hypothèse, ici, que lors de la saisie du courrier on a introduit le bon type de contenu et le bon département responsable (il est possible aussi que le type de contenu détermine le département concerné).

Messages d'entrée :

- Type-contenu
- Numéro-C.R.
- Date-réception
- Personne-concernée

Messages de sortie :

sortie : "Responsabilité transférée"

Effets sur le S.I. :

Ce transfert provoque :
- la modification d'une entité 'Contenu' (modification de l'attribut 'Personne-concernée').

i) Surveillance des délais (par journée).

Objectifs :

En fin de journée, on consulte les entités 'Contenu' et on dresse une liste de toutes celles qui ne sont pas encore traitées et qui normalement devraient être clôturées. On fait de même avec les entités 'Réponse'. Pour ces dernières, un rappel peut être expédié automatiquement au client.

Messages d'entrée :

Evénement "fin de journée"

Messages de sortie :

sortie 1 : une demande d'émission d'un rappel

sortie 2 : La liste des traces sélectionnées

Effets sur le S.I. :

La surveillance des délais provoque :
- la consultation des entités 'Contenu' et 'Réponse' (consultation des attributs 'Check-flag-contenu' et 'Date-ultime-suite' pour les 'Contenus' et des attributs 'Check-flag-réponse et 'Date-réception-réponse' pour les 'Réponses').

j) Suspendre, Annuler un courrier

Objectifs :

En fonction d'un événement interne ou externe, on suspend ou annule un courrier. Par exemple, la réception d'une lettre de décès peut entraîner la suppression de l'envoi d'un rappel éventuel.

Messages d'entrée :

- Numéro-C.R.
- Date-réception

Messages de sortie :

sortie : liste des traces modifiées ou supprimées

Effets sur le S.I. :

L'annulation d'un courrier provoque :
- la modification éventuelle d'une ou plusieurs occurrences des entités 'Contenu', 'Réponse', 'courrier-émis' et 'Courrier-reçu' (par exemple, en modifiant le Check-flag).

k) Localiser un courrier papier.

Objectifs :

En fonction de l'identifiant d'un courrier reçu, on visualise la localisation physique du courrier papier.

Messages d'entrée :

- Numéro-C.R.
- Date-réception

Messages de sortie :

sortie : Localisation-physique

Effets sur le S.I. :

La localisation physique du courrier papier provoque :
- la consultation d'une entités 'Courrier-reçu'.

1) Epuration de la base de données.

Objectifs :

Suivant des règles de gestion, on supprime la trace de courriers (reçus ou émis) de la base de données. Ces traces sont transférées sur un autre support d'accès non direct.

Messages d'entrée :

Une règle de gestion

Messages de sortie :

sortie 1 : La liste des traces sélectionnées

sortie 2 : Les traces sélectionnées sur un autre support

Effets sur le S.I. :

L'épuration provoque :
- la suppression de nombreuses occurrences de toutes les entités et associations à l'exception des entités 'Contrat' et 'Bibliothèque-Paragraphes',
- la modification éventuelle de l'entité 'Contrat' (modification de l'indicateur 'Archives').

m) Dire si un courrier est totalement traité.

Objectifs :

En fonction de l'identifiant d'un courrier (reçu ou émis), on indique s'il est totalement traité ou si celui-ci est en cours de traitement.

Messages d'entrée :

- Numéro-C.R. + Date-réception

- Numéro-C.E. + Date-émission

Messages de sortie :

sortie 1 : "le courrier est totalement traité"

sortie 2 : "le courrier est en cours de traitement"

Effets sur le S.I. :

Ce traitement provoque :
- la consultation de soit une entité 'Courrier-émis, soit une entité 'Courrier-reçu',
- la consultation éventuelle d'une ou plusieurs entités 'Réponse' ou 'Contenu' (si le Cleck-flag de entité 'courrier (émis/reçu)' indique que le courrier n'est pas totalement traité),

3.5. Exigences en sécurité

A ce stade du développement, on peut déjà relever certains points concernés par la sécurité.

Tout d'abord, le courrier expédié par un client doit être conservé sous sa forme papier. En cas de litige, il est le seul valable. Il faut donc prévoir un endroit "protégé" pour son stockage tant qu'il est utile à la gestion du contrat liant le client et l'entreprise.

Ensuite, l'accès aux données doit être réglementé dans le but de garder la confidentialité de la correspondance entre le client et l'entreprise. Pour cela, il est nécessaire d'identifier quels gestionnaires pourront accéder à quels traitements. Des protections internes à la base de données devront également être prévues.

Pour les services de gestion, toutes les données importantes se trouvent dans le système d'information, il faut donc prévoir des back-up réguliers et fréquents pour perturber le moins possible le fonctionnement de l'entreprise en cas de "disaster".

Dans le même ordre d'idées, comme les sorties papiers sont limitées, les données se trouvent souvent uniquement dans le système, il faut donc interdire la possibilité de supprimer des paragraphes de la bibliothèque et de modifier le contenu d'un courrier émis par l'entreprise lorsque celui-ci a été expédié au client. En effet, la demande de duplicata, de copies peut être exécutée à tout moment. Les données et les paragraphes composant le courrier doivent donc toujours être disponibles. De plus, la demande de duplicata doit être soumise à des contrôles strictes pour garder, entre autre, la confidentialité de la correspondance entre l'entreprise et ses clients.

Un autre type de sécurité concerne l'épuration de la base de donnée. Les données transférées peuvent toujours être consultées à l'occasion. Les supports d'archivage doivent donc être fiable et rester accessibles.

3.6. Préparation du test d'acceptation

Comme nous l'avions fait remarquer lors de la présentation de l'intégration de S.D.M. et des acquis méthodologiques de la C.G.E.R., la préparation du test d'acceptation ne peut être que très générale à ce stade. Celle-ci devra être complétée au fur et à mesure du développement.

Comme notre application est générale et comme nous n'avons aucune connaissance des activités de l'entreprise qui l'utilisera, il nous est impossible de donner une liste précise des tests à mener. Nous nous contenterons donc de quelques considérations générales.

Une première catégorie de tests à mener concernera les fonctionnalités du système. Les traitements identifiés et spécifiés devront s'exécuter correctement, sans erreur fonctionnelle.

Une deuxième catégorie de tests se rapportera aux aspects sécurité du système. Il sera utile de vérifier que les exigences dans ce domaine ont été prises en compte.

La convivialité du système sera également mise à l'épreuve, et ce, selon deux critères.
Premièrement, on testera les capacités d'aide et de guidage du système afin d'avoir une idée précise de ce que sera le temps nécessaire à l'apprentissage.
Deuxièmement, on s'assurera que le système répond bien aux critères de convivialité mis en évidence. Un système agréable à l'utilisation sera bien mieux accepté qu'un autre.

3.7. Analyse de l'impact organisationnel

Les réflexions menées ici restent également très générales. Néanmoins, elles concernent toute entreprise qui serait amenée à utiliser un tel système.

En ce qui concerne le personnel utilisateur, nous feront deux considérations.

Tout d'abord, à la question "sera-t-il utile d'engager du nouveau personnel ?", nous répondons : non. En effet, les gestionnaires actuels seront les futurs utilisateurs du système. Sauf cas exceptionnel, il est inopportun d'engager du personnel nouveau.

Ensuite, du côté de la formation, il sera nécessaire de prévoir des cours plus ou moins poussés suivant la complexité du système et l'habitude des futurs utilisateurs à manipuler un système informatique. Cette formation couvrira soit le sous-système de gestion du courrier, soit tout le système mis en place. La conception d'un manuel utilisateur extrêmement précis sera également nécessaire afin d'éviter toute confusion et tout sous-emploi du sous-système.

Les règles de gestion quant à elles (comment remplir les rubriques, quels sont les types permis pour une rubrique, etc.) devront aussi être précisées pour éviter au maximum les erreurs d'utilisation.

En ce qui concerne le matériel décentralisé, il faudra prévoir l'installation des éventuels nouveaux terminaux dans les différents départements qui recevront le nouveau système. Si l'implantation est progressive, un plan d'installation devra être mis au point.

Les nouvelles procédures de travail à définir seront plus ou moins variées suivant les organisations. Mentionnons simplement que pour ce qui est de l'envoi du courrier émis, il sera important de déléguer du personnel pour réceptionner le courrier à sa sortie du "centre d'impression" et pour le poster. Il faudra également prévoir un distributeur du courrier reçu aux différents départements.

3.8. Le matériel à acquérir

Un système tel que celui que nous étudions ne doit normalement pas être la première expérience informatique d'une entreprise. Un matériel relativement important devrait déjà être en place. Par contre, des terminaux spéciaux (format A4) pourront être envisagés. Ils faciliteront notamment la rédaction des courriers émis. De plus, des supports d'information supplémentaires

(disques par exemple) pourront être nécessaire afin de réserver un espace suffisant au système.

Un S.G.B.D. puissant remplacera peut-être le S.G.B.D. actuel si ce dernier ne donne pas de temps de réponse acceptables. Ce remplacement sera, à notre avis, exceptionnel mais pourra survenir vu les exigences élevées du système (nombreux points d'accès, volumes importants et opérations inter-fichiers fréquentes).

Chapitre 4

Conclusion

L'application présentée ci-dessus n'est pas vraiment une réalisation pratique au sens commun du terme. Elle se limite au étapes supérieures de l'analyse, et aucune implémentation n'a été réalisée, principalement faute de temps.

Elle nous a néanmoins permis de prendre conscience des avantages d'une bonne méthode supportée par des modèles adéquats.

De plus la solution proposée est utilisable dans de multiples environnements moyennant quelques adaptations et notamment l'intégration à d'éventuels systèmes déjà existants.

Le but recherché, à savoir l'emploi de la méthode étudiée, est, à notre avis, atteint à à peu près cinquante pour cent. Le développement complet nous aurait permis de cerner d'éventuels problèmes que nous n'avons pas identifié lors de la présentation théorique. Dès lors, seuls les utilisateurs réguliers de la méthode seront à même de la juger valablement.

PARTIE III

PRISE DE CONSCIENCE DES AVANTAGES ET DES LIMITES DES METHODES

Chapitre 1

Synthèse critique de la méthode S.D.M.

1.1. Introduction

Dans ce chapitre, nous tenterons d'établir quels sont les avantages et les inconvénients de la méthode S.D.M. ainsi que son domaine d'application. D'une part, nous examinerons comment S.D.M. traite le cycle de vie d'un projet de développement de logiciel. Pour cela, nous la comparerons à un cycle de vie qui nous semble représentatif pour la production de gros logiciels. D'autre part, nous regarderons si S.D.M. répond bien à l'attente des utilisateurs d'une démarche méthodologique.

1.2. Un cycle de vie d'un projet de développement [AVL]

1.2.1. L'analyse des besoins ou analyse d'opportunité

On définit un cadre général pour une solution automatisée, c'est-à-dire on produit une liste commentée des problèmes liés à la situation existante et des améliorations souhaitées; on donne une définition générale des objectifs du projet, ainsi qu'une première définition d'un modèle global pour l'application à développer; on fournit des informations de gestion de projet.

1.2.2. Spécification fonctionnelle du système informatisé

L'objectif de cette étape est d'obtenir une définition précise, complète et cohérente de ce que la solution automatisée devra réaliser (et pas comment elle le réalisera!), c'est-à-dire on produit une définition précise des fonctionnalités du système automatisé, des contraintes de performance, de l'environnement technique nécessaire à l'exploitation du système; on prépare le plan de tests d'acceptation. Cette définition est à la base du contrat liant le producteur du système au client.

1.2.3. La conception

Le but de la conception est d'obtenir une solution automatisée 'abstraite' qui réalise les spécifications développées au cours de l'étape précédente. Cette solution est faite d'algorithmes, de structures de données, et de différentes relations établies entre algorithmes, entre structures de données, et entre algorithmes et structures de données.

Le produit ainsi obtenu est exécutable par une machine abstraite, ce qui lui permet d'être multicible (transformable, de façon systématique, en un produit particularisé à une machine et un langage de programmation concrets).

Cette étape se subdivise généralement en deux sous-étapes successives : la conception globale et la conception détaillée.

- La conception globale.
  Obtenir une 'architecture logique' du système, c'est-à-dire une structuration de la solution automatisée en composants et en relations bien définies entre ces composants. Cette structuration garantira certaines propriétés (conformité aux spécifications énoncées lors des étapes précédentes, modifiabilité, réutilisabilité, portabilité, ...), et constituera la base de répartition du travail ultérieur entre programmeurs autonomes. Un composant dans l'architecture correspond, en effet, à une unité d'oeuvre associée à un traitement ou à une structure de données à développer ou à modifier. C'est l'étape critique dans le cycle de vie.

- La conception détaillée.
  Obtenir des solutions abstraites, en terme d'algorithmes

et de représentations de structures de données, qui réalisent les spécifications des différents modules de l'architecture logique.

1.2.4. Le codage

L'objet de cette étape est d'obtenir une solution automatisée 'concrète', implémentant la solution abstraite obtenue à l'étape précédente. Les programmes ainsi obtenus sont exécutables sur une configuration particulière matérielle/logicielle donnée.

Cette étape se subdivise généralement en deux sous-étapes : le codage global et le codage détaillé.

- Le codage global.
  Obtenir une 'architecture physique' du système, c'est-à-dire une structuration de la solution automatisée en composants physiques : unités d'exécution (programmes), unités de compilation (sous-programmes), procédures au sein d'un programme, ...

- Le codage détaillé.
  Produire, dans le langage de programmation cible, les différents modules de l'architecture physique par traduction et adaptation des modules correspondants obtenus à l'issue de la conception détaillée.

1.2.5. Conduite des tests, intégration et mise au point

Lors de cette étape, on cherche à détecter un maximum d'erreurs et défauts avant la livraison, par exécution du code sur base de plans de test, conçus pour la plupart au cours des étapes antérieures; on y remédie par retour en arrière dans la documentation et corrections (test d'acceptation, des programmes, de sécurité,

de la documentation, de performance, d'agrément à l'utilisation, d'installation).

1.2.6. La maintenance

On répercute, dans le logiciel produit et déjà exploité, un certain nombre de modifications qui peuvent être de nature fort différente :

1) Adaptations, extensions ou contractions du logiciel dictées par des modifications du même type dans les spécifications fonctionnelles, par des modifications dans l'environnement matériel/logiciel cible.

2) Améliorations eu égard aux performances, à l'agrément d'utilisation, ...

3) Corrections suite à la découverte tardive d'erreurs en cours d'exploitation.

1.3. Critique de la découpe en phases de S.D.M.

Dans le paragraphe précédent, nous avons présenté les étapes possibles d'un cycle de vie d'un projet de développement de logiciel. En vue d'une informatique de production, ce cycle de vie met en évidence plusieurs points :

- Est-il nécessaire de développer un projet ?
- Que devra exécuter le système d'information ?
- Comment seront réalisées les fonctionnalités de l'application sur une machine 'abstraite' ?
- Adaptation de la solution conçue à une machine et un langage de programmation.
- La solution développée est-elle correcte ?
- Est-il utile de continuer à maintenir le système ?

La découpe de S.D.M. privilégie également ces différents points. Toutefois, il existe des différences entre les deux découpes.

Le but de la phase 1 -analyse de l'existant et analyse conceptuelle- est de formuler une expression claire et non ambiguë des besoins. Celui de la première étape du cycle de vie -analyse d'opportunité- est identique.

le but de la phase 2 -analyse fonctionnelle- et celui de l'étape -spécification fonctionnelle du système informatisé- sont équivalents, c'est-à-dire définir précisément toutes les spécificités du système. Toutefois, S.D.M. va plus loin dans l'analyse. La découpe en sous-systèmes, en d'autres mots une partie de l'architecture logique, est déjà réalisée au cours de cette étape. Une autre distinction : S.D.M. n'aborde pas le test d'acceptation qui se conçoit sur base des fonctionnalités de l'application précisées dans cette phase.

La phase 3 -analyse technique- qui correspond à l'étape de conception se veut, aussi, indépendante d'un environnement logiciel et matériel mais est moins complète. L'importance de la modularité de la solution n'est pas mise en évidence et l'architecture logique entamée lors de la phase précédente n'est pas approfondie. S.D.M. ne privilégie pas cette phase qui est considérée comme critique surtout pour la maintenance ultérieure.

La phase 4 -réalisation des programmes- envisage sommairement le problème de l'architecture physique de la solution automatisée élaborée lors de l'étape de codage. Elle s'intéresse principalement à la description et à la traduction des programmes.

La phase 5 -tests- qui correspond à l'étape -conduite des tests, intégration et mise au point- devrait être préparée beaucoup plus tôt : le test d'acceptation se rapportant aux fonctionnalités du système, les tests d'intégration des programmes en fonction des spécifications des modules et des relations entre modules.

La phase 6 -conversion et mise en production- ne figure pas dans le cycle de vie d'un logiciel. Elle s'occupe de

problèmes annexes aux phases 3, 4 et 5. Son contenu est inclus dans les étapes de conception, de codage et de tests.

La phase 7 -maintenance- poursuit les mêmes objectifs que la dernière étape du cycle de vie.

On remarque, donc, que S.D.M. s'occupe de tout le cycle de vie d'un système d'information. S.D.M. met l'accent sur l'importance des deux premières phases -étude de l'existant, analyse conceptuelle, analyse fonctionnelle- alors que toutes ont leur importance. L'étape de conception réalise l'architecture logique qui facilite la maintenance ultérieure. L'étape de codage explicite les relations entre programmes. L'étape de tests permet de détecter un maximum d'erreurs avant l'exploitation.

1.4. Critique de la démarche méthodologique de S.D.M.

"Une méthode est une démarche, généralement structurée en étapes, qui permet d'analyser et de concevoir les systèmes d'information en utilisant des modèles, des langages et des outils (automatisés)" [CASTELLANI, p.5].

Cette définition est généralement admise pour décrire une méthodologie de développement de systèmes d'information. Dans le prochain chapitre, nous reviendrons plus longuement sur l'utilité d'une démarche méthodologique, et les éléments qui la composent, pour produire des logiciels de qualité, modifiables à faible coût.

Dans ce paragraphe, nous examinons si S.D.M. utilise tous les éléments qui constituent cette définition.

- Reposer sur des modèles, des techniques :

"Un modèle est un ensemble de concepts, de règles et de conventions de représentation (graphiques en particulier) qui permettent de représenter des phénomènes" [CASTELLANI, p.4]. Ces modèles augmentent la communicabilité, la

précision, la standardisation des documents. S.D.M., se voulant indépendante de techniques de réalisation, n'en propose que très peu (schémas E/A, IPO, DFD, tables de décision). Ces techniques concernent essentiellement les deux premières phases.

- Disposer d'outils, d'un langage :

Les outils sont des instruments que les analystes et les concepteurs utilisent pour la spécification, la documentation, le contrôle, la gestion (dictionnaire de données, générateurs, documenteurs, dossiers standards). "Un langage est constitué d'un ensemble de signes, de mots et de règles d'agencement de ces signes et de ces mots permettant de représenter et d'utiliser les phénomènes gérés par les systèmes d'information" [CASTELLANI, p.4]. S.D.M. propose deux type d'outils, les dictionnaires de données et le prototypage. Cette limitation découle de l'indépendance vis-à-vis des techniques que S.D.M. désire garder.

- Règles de mise en oeuvre :

Une démarche, généralement structurée en étapes, donne un inventaire de toutes les tâches à entreprendre pour atteindre un but bien défini : produire du logiciel de qualité à un coût prévisible. S.D.M. propose une liste d'activités structurée en sept étapes. Cette liste est détaillée et s'intéresse à tous les aspects du développement. C'est l'un des avantages de cette méthode.

S.D.M. ne fait donc aucun présupposé sur l'environnement de développement existant dans l'entreprise qui la choisit. La méthode n'impose aucun modèle, outil, langage et s'adresse, par conséquent, à un large public. Les points actuels de contrôle sont basés sur les documents produits après chaque activité ou chaque phase. Les responsables de ces contrôles ne sont pas identifiés par la méthode.

1.5. Synthèse des avantages et inconvénients de S.D.M.

"Les organisations conçoivent, réalisent et utilisent des systèmes d'information pour satisfaire les besoins en information engendrés par leurs comportements organisationnels" [BOPI, p.1]. Le système d'information met en oeuvre l'ensemble des moyens (humains, techniques, budgétaires, organisationnels, de documentation, ...) nécessaires au traitement et à l'exploitation des informations dans le cadre d'objectifs définis. S.D.M. s'intéresse donc bien à tous les éléments concernés par le développement d'un projet.

Ces éléments sont repris dans une liste d'activités proposée par la méthode. Un des mérites de S.D.M. est cette liste complète, bien découpée, adaptable suivant le projet à développer. Mais les plannings proposés ne donnent pas une vue d'ensemble de l'enchaînement de toutes les activités. On peut résoudre ce problème lorsque l'on adapte la méthode à un site de développement particulier. D'autre part, certaines activités peuvent être réalisées beaucoup plus tôt dans le développement et donc reprises des phases antérieures (par exemple, la préparation du test d'acceptation peut débuter dès l'analyse fonctionnelle). Un autre point négatif, chaque activité est mise sur pied d'égalité et on ne distingue pas forcément celles que l'on pourrait retarder en cas de problèmes. Le contenu de différentes activités est difficilement évaluable et contrôlable : en effet des termes tels que 'global', 'en général' ne donnent pas une limite précise du contenu d'une activité.

Les outils et modèles ne sont pas inclus ou énumérés par la méthode or ceux-ci permettent de gagner du temps et facilitent la compréhension et la propreté des résultats. Une liste d'outils et modèles devrait être proposée (cfr. partie I chapitre 3). Néanmoins, le choix de cette méthode a l'avantage de ne rien imposer aux développeurs et

donne donc la possibilité de reprendre les outils existants.

Une autre possibilité pour gagner du temps est d'employer au mieux les compétences des personnes pour la réalisation des activités. Les personnes nécessaires à la réalisation des différentes activités ne sont pas mentionnées, l'énoncé du contenu des activités indiquant le type de personnes utiles. Toutefois, une liste de personnes impliquées et leur responsabilité permettrait de mieux gérer les compétences.

A la base du contrôle possible, la documentation est présente dans chaque activité mais n'est pas formalisée. La mise en oeuvre de la méthode devrait être accompagnée d'une description de standards de la documentation car celle-ci est très importante.

> Remarque : Toute documentation est importante car elle permet :
> - d'orienter et d'accélérer le travail de réflexion, la coordination entre les différents groupes,
> - de diminuer la vulnérabilité du projet et du groupe d'étude,
> - de simplifier le travail par des standards,
> - de faciliter ultérieurement la maintenance,
> - d'utiliser efficacement le système après sa mise en production.

La gestion du projet est facilitée en utilisant S.D.M. : il est possible d'élaborer un planning général de développement et un système de contrôle de complétude en utilisant la liste des activités. A un moment donné, il est possible de contrôler que toutes les activités, devant être terminées, sont effectivement réalisées.

S.D.M. est destinée aux grosses entreprises pour uniformiser, sécuriser la gestion des projets. En effet, l'application de la méthode, à la lettre, est contraignante. S.D.M. est surtout intéressante pour le développement de nombreux projets de grandes envergures.

Le but d'une méthode étant de développer et gérer correctement des systèmes d'information, S.D.M. offre une aide pour uniformiser les développements, mais devrait être complétée, à notre avis, pour développer des projets informatiques rapidement et correctement. Des outils et des techniques, ajoutés à la méthode, faciliteraient le travail d'analyse et de conception des développeurs. S.D.M. est une méthode suffisamment souple pour permettre de garder les acquis méthodologiques tout en garantissant plus de rigueur tout au long du développement.

Chapitre 2

Conclusions

Au cours des dernières décennies, l'importance accordée à l'information au sein de l'entreprise n'a cessé de s'accentuer. A côté de ce phénomène, on a également assisté à une augmentation sensible des volumes qui a rendu la gestion de cette information de plus en plus complexe.

Les organisations en étaient arrivées à un point de rupture où il ne leur était plus possible de traiter manuellement leurs systèmes d'information sans nécessiter de restructuration profonde.

L'introduction de systèmes d'information automatisés allait apporter une solution à ces deux problèmes. Grâce à elle, il devenait possible de gérer des quantités croissantes d'informations avec une efficacité et une rigueur maximales.

Pourtant, d'autres problèmes liés à l'automatisation allaient surgir. Les systèmes d'information automatisés étant un nouveau concept, rien de précis concernant leur mise en oeuvre n'était établi et leur développement fut anarchique. Le manque total de standardisation et d'intégration a mis alors en évidence la nécessité d'utiliser des méthodes pour le développement de ces nouveaux produits.

A l'heure actuelle, nombreuses sont les méthodes qui proposent une démarche pour aboutir à une bonne solution. Malheureusement, leur emploi n'est pas toujours des meilleurs et ce, pour plusieurs raisons que nous avons déjà évoquées au cours de ce mémoire et que nous allons reprendre brièvement ci-dessous.

Pour être efficace, une méthode devrait reposer sur des techniques de modélisation et être mise en oeuvre à l'aide d'outils logiciels[52]. Les modèles décrivant le système devraient être construits suivant des règles précises, et les outils faciliteraient leur élaboration en vérifiant

---

52. Inspiré de [BOPI] et de [ROLL].

notamment leur cohérence, leur complétude, ou encore leur adéquation. La méthode serait, quant à elle, composée de plusieurs étapes indiquant chacune comment utiliser tel modèle ou tel outil afin d'aboutir à une solution intermédiaire ou à la solution finale[53].

Si des méthodes spécialement prévues pour les systèmes d'information modernes ont été mises au point, comment se fait-il qu'il subsiste encore autant de problèmes dans les entreprises, que ce soit au niveau du développement ou au niveau de la maintenance ?

Pour apporter une réponse valable à cette question, des dizaines d'années d'expérience seraient nécessaires. Néanmoins, nous pouvons mettre en avant deux considérations qui nous semblent pertinentes.

La première est que les développeurs ont trop souvent tendance à exiger d'une méthode d'être une panacée qu'il suffit de détenir pour mettre au point des systèmes performants. Une méthode serait alors comparable à un toboggan sur lequel il suffit de se laisser glisser pour arriver en bas. Toute chute est pratiquement impossible.

Cette vision optimiste est malheureusement erronée. La réflexion humaine, la démarche philosophique, restent le propre de l'homme. Aucune méthode ne pourra les remplacer. Avant de se lancer dans l'élaboration d'un système, il faut un but. Le méthode ne sera qu'un support (grâce à une standardisation se traduisant par l'emploi d'un langage commun, d'une démarche commune, etc.) qui guidera les développeurs dans leur travail, mais elle ne trouvera jamais la solution à leur place.

La seconde considération que nous tenons à mettre en évidence est que les méthodes sont généralement mal employées dans la plupart des entreprises, surtout parce que

---

53. Une solution intermédiaire pourrait être un schéma E-A, ou une description de programmes.

leurs champs d'application et leurs limites ne sont pas explicités à leurs utilisateurs. Ces entreprises sont en constante évolution et les logiciels développés doivent avoir une grande souplesse afin de suivre sans trop de peine cette évolution. Leur réalisation est possible soit si les développeurs réfléchissent dans ce sens lors de l'analyse, soit s'ils utilisent une méthode particulière aidant à la conception de pareils logiciels. Malheureusement, de telles méthodes sont généralement plus complexes que les méthodes traditionnelles et sont donc beaucoup plus difficiles à introduire.

Mais, dès l'instant où la méthode n'est plus (ou mal) suivie le désordre risque de s'installer, entraînant ainsi la production de logiciels très moyens.

Comme nous l'avons déjà évoqué dans le chapitre consacré aux modèles et aux outils[54], les raisons de leur non-emploi sont variées. Parmi celles-ci figurent au premier rang la peur du contrôle hiérarchique, la peur de perte d'un statut d'indispensabilité, mais aussi l'idée que les méthodes ainsi que les modèles et outils sous-jacents sont inutiles.

Toutes ces considérations nous amènent à nous poser deux questions : quel est le profil de la méthode idéale et comment la faire accepter par les développeurs ?

Une réponse simpliste à la première question consisterait à dire que, étant donné la variété importante des environnements et l'incessante évolution, il n'existe et il n'existera sans doute jamais de méthode idéale. Nous voudrions néanmoins mettre en avant quelques aspects importants qu'aucune approche ne devrait ignorer.

L'idée de continuité est, à notre avis, primordiale. Non seulement, il est indispensable qu'une méthode couvre tous les aspects du développement, y compris la maintenance, mais

---

54. Cfr. supra partie I, chapitre 3.

elle doit également permettre de passer d'un stade à l'autre avec un maximum d'aisance. L'idée de raffinements successifs est particulièrement bien adaptée ici. Elle ramènerait le développement à une suite de spécifications de plus en plus fines permettant, à partir d'un problème initial, d'aboutir à une solution. Cependant, pour que cette approche soit valable, il est nécessaire de savoir dès le début de l'analyse ce que l'on veut construire et de le spécifier complètement. L'importance de la phase conceptuelle est soulignée ici car le travail effectué lors de celle-ci est évalué directement, alors que dans les autres approches, la négligence de cette phase n'est ressentie que plus tard dans la développement (programmation, maintenance).

Un autre aspect non négligeable des méthodes est qu'elles doivent fournir des moyens efficaces d'évaluation et de contrôle du travail accompli ainsi que des décisions prises au cours de l'analyse. Ces évaluations doivent être fréquentes afin d'éviter au maximum les "backtracking". Rappelons-nous que l'absence de points de contrôle dans la méthode initiale de la C.G.E.R.[55] constituait un sérieux handicap pour les responsables du M.A.I. Dans l'intégration que nous avons proposée[56], les treize points de passage nous ont semblé offrir un bon moyen de vérification et de validation d'un projet.

Un aspect souvent négligé par les méthodologies est l'estimation du temps nécessaire au développement ainsi que l'évaluation des coûts. Une remarque fréquente des développeurs que nous avons eu l'occasion de côtoyer était qu'ils n'utilisaient pas complètement une méthode faute de temps. L'élaboration d'un système d'information est, en effet, généralement soumise à de fortes contraintes de temps et d'argent. Si une approche maîtrisait parfaitement ces

---

55. Cfr. supra partie I, chapitre 1.

56. Cfr. supra partie I, chapitre 4.

deux variables, ou du moins si elle permettait de les évaluer avec une bonne précision, un réajustement des projets (par exemple par une réduction de leur ampleur) rendrait l'emploi d'une méthode possible aux yeux des développeurs et améliorerait sans doute la qualité des logiciels produits.

Au-delà de ces considérations, une question épineuse reste en suspens. Quel degré de liberté une approche doit-elle laisser aux développeurs ?

Il existe deux grandes classes de méthodes.

La première est celle des méthodes de type "fil conducteur", dans laquelle on retrouve par exemple S.D.M.[57], et qui se contentent de mettre en évidence ce qu'il convient de faire au cours de l'analyse, mais qui ne donnent aucun moyen pour le réaliser. Cependant, la limite entre le "quoi faire" et le "comment faire" n'est pas nette, et de telles méthodes devront quand même expliquer aux développeurs comment réaliser le quoi, même si elles ne donnent aucune technique.

La seconde regroupe les méthodes que l'on pourrait appeler intégrées. Celles-ci offrent des techniques de réalisation et parfois des outils logiciels pour les mettre en oeuvre.

Sans trop entrer dans les détails, la première catégorie s'adapte bien à des environnements partiellement définis comme celui que nous avons rencontré à la C.G.E.R. Elle permet de compléter les acquis sans les éliminer[58]. Par contre, du fait de leur aspect "checklist", ce type d'approches nous parait limité si aucun élément méthodologique n'est en place.

La seconde catégorie, quoiqu'étant beaucoup plus contraignante, nous paraît être la meilleure. Au-delà d'une

---

57. Cfr. supra partie I, chapitre 2, partie III, chapitre 1.

58. Cfr. supra chapitres 1 et 4.

démarche, ce type de méthodes propose, ou plutôt impose, une série de modèles et parfois certains outils pour assister les concepteurs dans leur travail d'analyse. La présence de techniques standardisées, facilitant entre autre la communication, autorisera plus facilement le travail en équipe. Le temps supplémentaire consacré à la rédaction de la documentation au cours de l'analyse n'est en aucun cas perdu. Une bonne analyse conduira au développement de bons logiciels adaptés aux besoins et nécessitant un minimum de maintenance.

Cependant, la présence de ces techniques ne devra en aucun cas constituer un frein pour les développeurs.

En ce qui concerne les outils logiciels, nous avons vu quelles étaient leurs dangers et leur limites[59]. Pourtant, nous sommes favorables à l'idée de dériver une solution à partir des spécifications. Toutefois, il nous semble qu'il serait plus honnête de ne plus parler de spécifications mais de langage de très haut niveau, non accessible à tous.

L'avantage de ce type d'outils serait qu'une modification dans les spécifications générerait automatiquement une nouvelle solution avec un risque normalement nul d'erreur. Dans une telle perspective, on peut considérer que La productivité des informaticiens devrait être considérablement améliorée. Malheureusement, ces outils n'en sont qu'à leurs premiers balbutiements et il nous semble qu'il serait plus intéressant d'attendre leur propagation avant d'entamer un débat à leur sujet.

Il nous reste maintenant à essayer de répondre à notre deuxième question : comment faire accepter les méthodes ?

Une solution à écarter d'emblée consisterait à les imposer de manière autoritaire. Dans ce cas, les réactions des développeurs seraient vraisemblablement négatives. Leur

---

59. Cfr. supra partie I, chapitre 3.

résistance se manifesterait sous la forme d'une utilisation impropre de la méthode qui rendrait leur travail médiocre. Ils accuseraient ensuite cette méthode en affirmant qu'elle est inadéquate.

Selon nous, une bonne solution serait de former les concepteurs à l'utilisation des méthodes, des modèles et des outils. Il est non seulement utile de leur procurer une démarche, mais il faut également leur montrer comment l'appliquer. De plus, il est nécessaire de parvenir à implanter dans leur esprit le sentiment que, sans une bonne approche, un travail soigné est pratiquement impossible.

Au-delà de ce problème d'introduction des méthodes subsiste celui de la formation à leur utilisation. Si l'on se contente de fournir les techniques aux développeurs et même s'ils les acceptent, la tâche la plus importante reste de leur montrer la bonne façon de s'en servir.

C'est à notre avis au niveau des outils de spécification que le danger de mauvais emploi est le plus présent. Comme nous l'avons fait remarquer lors de leur analyse, leurs utilisateurs risquent de les considérer comme des assistants intelligents, voir même comme des remplaçants. Or, ils n'en sont pas et n'en seront sans doute jamais. Dès lors, une solution acceptable serait d'amener les concepteurs à travailler d'abord comme ces outils, afin qu'ils découvrent les mécanismes sous-jacents à leur fonctionnement (c'est-à-dire découvrir comment ils travaillent), et qu'ensuite, on les leur procure pour qu'ils puissent en tirer un profit maximum. La connaissance des principes de fonctionnement ferait prendre conscience aux développeurs des limites et des dangers de ce type d'outils et éliminerait, à notre avis, une bonne partie des risques de mauvaise utilisation.

Notre manque d'expérience nous limite quelque peu dans notre activité de critique. Pourtant, les quelques points mis en évidence dans ce chapitre nous semblent pouvoir influencer fortement la qualité des systèmes d'informations modernes.

Au cours de ce mémoire, nous avons présenté une méthode de développement particulière et nous en avons fait la critique. Nous avons vu quels étaient ses points forts mais aussi quelles étaient ses limites.

Nous nous sommes également questionnés sur les possibilités d'intégration de cette méthode dans un environnement partiellement défini et nous avons examiné pourquoi elle constituait le meilleur compromis.

Pour réaliser notre intégration, nous avons également pris en compte les différents modèles et outils disponibles et nous nous sommes interrogés sur la manière de les mettre en oeuvre. Nous avons découvert que, pour certains outils, des dangers étaient présents (par exemple l'utilisation à outrance d'un outil de prototypage) et que leur utilisation devait être faite avec la plus grande prudence.

Comme il nous paraissait impossible de réaliser une critique de la méthode sans l'utiliser nous même, nous avons réalisé une partie d'une application en utilisant l'intégration que nous avons proposée. Cette application nous a permis de mieux comprendre les points de vue des différents acteurs en présence lors du développement d'un nouveau système d'information. Il est pourtant regrettable que nous n'ayons pas pu travailler en commun avec le groupe d'étude. Ce manque de collaboration a rendu notre étude plus difficile et nous a peut-être fait passer à côté d'autres points essentiels.

Malgré nos tentatives de réponse, plusieurs questions restent en suspens au terme de ce travail : une méthode est-elle réellement efficace ? Existera-t-il un jour une méthode universelle ou sera-t-il toujours nécessaire de "customiser" une approche générale ? La réalisation des systèmes d'information sera-t-elle un jour uniquement l'oeuvre de logiciels spécialisés ?

Toutes ces questions trouveront peut-être une réponse dans les années à venir, mais d'ici là, restons lucides et gardons-nous bien d'anticiper.

Liste des sigles utilisés

C.T.I. : "Centre de Traitement de l'Information".

D.B.A. : "Data Base Administrator" ou administrateur des bases de données

D.F.D. : "Diagramme des Flux de Données". Représentation de la circulation des informations dans l'organisation au travers des différents traitements.

D.M.R. : "Data Manager". Dictionnaire de données utilisé à la C.G.E.R.

D.S.I. : "Développement des Systèmes d'Information".

E-A : "Entité-Association".

I.P.O. : "Input Process Output". Représentation d'un traitement en décrivant ses entrées, le processus de transformation des entrées en sorties, et les sorties.

J.S.P. : "Jackson Structured Programming". Technique de structuration des flux de données et des programmes.

M.A.G. : "Modèle d'Accès Généralisé".

M.C.D. : "Modèle Conceptuel des Données". Représentation des objets et des relations inter-objets d'un système sous forme de schéma Entité-Relation (ou Entité-Association).

M.C.T. : "Modèle Conceptuel des Traitements". Représentation des traitements au moyen de plusieurs modèles (S.H.A.T., I.P.O. et D.F.D.).

PALOMA : "PAth LOad MAtrix". Logiciel de validation mutuelle des modèles conceptuels des données et des traitements sur base du contenu de DATAMANAGER.

S.A.D.T. : "Strutured Analysis and Design Technique".

S.D.M. : "System Development Methodology". Méthode de développement de systèmes d'information introduite à la CGER.

S.G.B.D. : "Système de Gestion de Base de Données".

S.H.A.T. : "Structure Hiérarchique des Activités et Tâches". Approche "top-down" d'un système par décomposition de plus en plus fine aboutissant à des opérations élémentaires pour l'utilisateur.

S.I. : "Système d'Information".

Bibliographie

[AVL] : J. HOORENS, P. SCHOENMAEKERS, A. VAN LAMSWEERDE : "Les contrats de développement de logiciels sur mesure", Précis de la faculté de droit de Namur, 5, 1986. Centre de Recherche Informatique et Droit des Facultés Universitaires de Namur, Story Scientia Bruxelles

[BOPI] : F. BODART, Y. PIGNEUR : "Conception assistée des application informatiques, 1 : étude d'opportunité et analyse conceptuelle", MASSON, PUN, 1983-1989.

[CASTELLANI] : X. CASTELLANI : "Méthode générale d'analyse d'une application informatique", tome 0 : "Macro-analyse, étude préalable et analyse conceptuelle des systèmes d'information", MASSON, 1987

[CODD] : CODD : "A relationnal Model of data for large, shared, Data Banks", com. of the ACM, juin 1970.

[DATA] : Data Decision, n° 89, 16 novembre 1988.

[DEDE] : N. DELIEUX, B. DELVAUX : "Etude d'une méthode et d'outils d'aide au développement d'application de gestion", mémoire de fin d'étude, FNDP 1987.

[DMR] : Notes sur le dictionnaire de données Data Manager.

[DOL] : Premier cours SDM donné à la CGER par la société DOLMEN sur la méthode SDM.

[EXCEL] : "EXCELERATOR", brochure distribuée par Index Technology; Index Technology, 1987.

[FAIRLY] : R. FAIRLY : "Software Engineering Concepts", MacGRAW HILL International Edition, computer services, 1985.

[GRADY] : E. de GRADY : "Etude de la méthode de programmation structurée de JACKSON et tentative de formalisation", mémoire de fin d'étude, FNDP 1988.

[HAI 1] : J.-L. HAINAUT : "Introduction à la théorie relationnelle des bases de données", FNDP, mars 1988.

[HAI 2] : J.-L. HAINAUT : "Conception assistée des applications informatiques, 2 : conception de la base de données", MASSON, PUN, 1986.

[IMS] : B. DELVAUX : "Guidance pour le développement de projets en I.M.S.", C.G.E.R., septembre 1988

[JACK 1] : M. JACKSON : "System development", Prentice/Hall International, 1983.

[JACK 2] : M. JACKSON : "Principle of Program Design", Academic Press Inc. Ltd, 1975.

[LUYCKX] : W. LUYCKX : "Manuel pour l'utilisation du dictionnaire de données DataManager", C.G.E.R., Bruxelles.

[MAI 1] : C. GRAAS, A. SPILTOIR, R. VAN'T DACK : "Modélisation conceptuelle", doc. M.A.I., septembre 1988.

[MAI 2] : C. GRAAS : "Analyse conceptuelle des données", doc. M.A.I., décembre 1987.

[MAI 3] : C. GRAAS : "Modélisation des traitements", doc. M.A.I.

[MAI 4] : C. GRAAS : "Introduction à la méthodologie", doc. M.A.I.

[MAI 5] : C. GRAAS : "Normalisation"; doc. M.A.I.

[MAI 6] : C. GRAAS : "Définitions utilisées lors de l'analyse conceptuelle", V1.0, CGER, avril 1988

[MAI 7] : C. GRAAS, R. VAN'T DACK : "Analyse des spécifications", C.G.E.R., septembre 1988

[MAI 8] : "Répartition des tâches entre analystes et programmeurs dans un environnement J.S.P.", document M.A.I.

[MAT] : I. MATHIEU : "Conception et réalisation des applications de gestion : une approche basée sur l'explication des raisonnements", mémoire de fin d'étude, FNDP 1986.

[METZNER] : J.R. METZNER, B.H. BARNES : "Decision Tables Languages and Systems", ACM Monograph series, Academic Press Inc., 1977.

[ROLL] : C. ROLLAND, O. FOUCAUT, G. BENCI : "Conception des systèmes d'information : la méthode REMORA".

[SDM] : H.B. EILERS : "Systeem ontwikkeling volgens SDM", academic service 1987, Den Haag.

[SDM-ANG] : H.B. EILERS, W.S. TURNER, R.P. LANGERHORST, G.F.HICE, A.A. UIJTTENBROEK : "SDM - SYSTEM DEVELOPMENT METHODOLOGY", Elsevier science publishers B.V., 1987

[TELON] : "TELON, System Overview", brochure distribuée par Pansophic; Pansophic Systems Inc., 1988.

CONTRIBUTION A L'INTRODUCTION D'UNE METHODOLOGIE DE DEVELOPPEMENT DANS UNE ENTREPRISE

ANNEXES

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

Promoteur : **M. Baudouin LE CHARLIER.**

Année académique : **1988-1989.**

Table des Annexes

Annexe A : La Demande de Développement

Annexe B : Aperçu synthétique de la méthode de développement de logiciels de la C.G.E.R.

Annexe C : Elaboration d'un Modèle Conceptuel des Données

Annexe D : La Normalisation

Annexe E : Le Diagramme des Flux de Données

Annexe F : La Structuration d'un système

Annexe G : Le Diagramme I.P.O.

Annexe H : Construction d'un algorithme en Langage Structuré

Annexe I : Les Tables de Décision

Annexe J : Les éléments de base de J.S.P.

Annexe K : Configuration de DataManager à la C.G.E.R.

Annexe L : Schéma détaillé des activités

Annexe M : Schéma par raffinements successifs des activités

Annexe N : Les références aux activités de S.D.M.

Annexe A

La Demande de Développement

```
====================
|      C.G.E.R.    |
|                  |
|      C.T.I.      |
====================
```

```
============================
| DEMANDE DE DEVELOPPEMENT |
============================
```

version 1.6 du 14/07/88
MAI - 061.6

==========

C.G.E.R.		p.2
	DEMANDE DE DEVELOPPEMENT	
C.T.I.		version 1.6

AVANT-PROPOS

Ce formulaire vient en aide aux utilisateurs de l'informatique et permet au CTI d'apprécier plus aisément les demandes qui lui sont soumises.

Aussi le CTI se tient-il à la disposition de ces utilisateurs pour remplir le formulaire avec eux.

Quand remplir le formulaire ?

Lors d'une demande
- d'un nouveau système,
- d'un code d'application
  (code de deux lettres, conforme aux conventions de noms, demandé avec le formulaire "application code assignment form"),
- 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 d haque 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.

A qui le renvoyer ?

L'exemplaire original est adressé à l'Informaticien D concerné par la demande de développement, tandis qu'une copie est adressée au service MAI, groupe Méthodes et Formation.

MERCI D'AVANCE.

C.G.E.R.	DEMANDE DE DEVELOPPEMENT	p.3
C.T.I.		version 1.6

1. GENERALITES.

DATE DE REDACTION :

CODE D'APPLICATION (si connu) : . .

INTITULE DU DEVELOPPEMENT :

SERVICE(S) CONCERNE(S) :

No(s)

NCTIONNAIRE RESPONSABLE : NOM :
DE LA DEMANDE

TEL :

SIGNATURE :

MANAGER UTILISATEUR : NOM :
(Rang 13 minimum)

TEL :

SIGNATURE :

CODE(S) PPBS DU(DES) SERVICE(S) UTILISATEUR(S) ET REPARTITION DES COUTS

Code	%	Service

LISTE DES MEMBRES DU GROUPE DE TRAVAIL

C.G.E.R.	DEMANDE DE DEVELOPPEMENT	p. 4a
C.T.I.	Indications de rédaction	version 1.6

. 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... .

. CAUSES ET ORIGINE DU PROBLEME.

Mentionnez par exemple les problèmes concernant les couts, 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.

C.G.E.R.	DEMANDE DE DEVELOPPEMENT	p. 5a
C.T.I.	Indications de rédaction	version 1.6

. 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.

Annexe B

Aperçu synthétique de la méthode de développement de logiciels de la C.G.E.R.

	DONNEES	TRAITEMENTS	
	DEMANDE DE DEVELOPPEMENT		
ANALYSE CONCEPTUELLE	- Elaboration d'un modele conceptuel des donnees en utilisant la technique Entite - Association. - Quantification du modele conceptuel des donnees (ajout des estimations des volumes) => modele statique des donnees.	- Structuration du systeme avec la S.H.A.T. - Construction d'un diagramme des flux de donnees (D.F.D.) - Description des taches via un modele I.P.O.	QUOI
	PALOMA		
ANALYSE FONCTIONNELLE	- Normalisation (selon les trois premieres formes normales principalement). - Modele dynamique des donnees construit en utilisant les resultats fournis par PALOMA. - Conception du modele logique des donnees (identification des hierarchies (IMS) et des relations (DB2)) a partir du modele dynamique des donnees.	- Identification des taches ON-LINE et des des taches BATCH, des taches manuelles et des taches automatisees. - Description des taches via un I.P.O. auquel on ajoute l'origine des entrees et le destination des sorties, et dans lequel on defini les traitements.	COMMENT
ANALYSE TECHNIQUE	- Definition physique de la base de donnees.	- Description des programmes et des modules.	COMMENT DU COMMENT
PROGRAMMATION		- Utilisation de la methode JACKSON.	

Annexe C

Elaboration d'un Modèle Conceptuel des Données

L'exemple que nous présentons ici est issu d'un cours donné par le M.A.I. Il met en oeuvre un système simplifié de distribution de livres.

1. Liste brute des informations

Une première étude des informations à traiter conduit à la liste suivante :

- numéro du livre
- nom du livre
- nom de l'auteur
- nom de l'éditeur
- nom du dépôt
- numéro de téléphone
- nom de l'éditeur
- quantité en stock
- numéro de l'ouvrage
- adresse de l'éditeur

Cette liste contient des redondances. La liste épurée les élimine et définit chaque élément.

2. Liste épurée des informations

- numéro du livre : numéro désignant univoquement l'ouvrage.
- nom du livre : titre du livre
- nom de l'auteur : nom et initial des prénoms du ou des auteurs du livre
- nom de l'éditeur : nom de la maison d'édition
- nom du dépôt : nom de l'endroit de dépôt des livres
- numéro de téléphone du dépôt
- numéro de téléphone de l'éditeur
- quantité en stock : nombre d'exemplaires d'un ouvrage, imprimés chez un certain éditeur, et disponibles dans un dépôt donné

3. Les types d'entités

- LIVRE (identifiant : numéro du livre)
- AUTEUR (identifiant : nom de l'auteur + initiales des prénoms)
- EDITEUR (identifiant : nom de l'éditeur)
- DEPOT (identifiant : nom du dépôt)

4. Rattacher leurs propriétés aux entités

- LIVRE numéro du livre, nom du livre
- AUTEUR nom de l'auteur + initiales des prénoms
- EDITEUR nom de l'éditeur, adresse de l'éditeur, numéro de téléphone de l'éditeur
- DEPOT nom du dépôt, numéro de téléphone de dépôt

5. Définitions des associations

- une association binaire "ECRIRE" relie LIVRE et AUTEUR : un auteur écrit un livre et un livre est écrit par un auteur.

- une association binaire "PARAITRE" relie LIVRE et EDITEUR : un livre paraît chez un éditeur.

- une association tertiaire "STOCKER" relie LIVRE, EDITEUR et DEPOT : un éditeur d'un livre le stocke dans un dépôt.

L'association STOCKER est porteuse; son attribut est "quantité en stock".

6. Déterminer les cardinalités

- On ne considère que les ouvrages édités chez au moins un éditeur, mais un ouvrage peut paraître chez plusieurs éditeurs. Les cardinalités de "LIVRE - PARAITRE" sont donc (1,n).

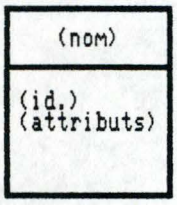
- On ne considère également dans le système que les éditeurs dont on distribue au moins un livre. Les cardinalités de "EDITEUR - PARAITRE" sont donc (1,n).

- Comme il existe des ouvrages anonymes et qu'un ouvrage peut être écrit par plusieurs auteurs, les cardinalités de "LIVRE - ECRIRE" sont (0,n).

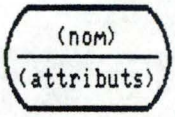
- Un auteur écrit au moins un livre. Les cardinalités de "ECRIRE - AUTEUR" sont donc (1,n).

- Un livre est stocké dans au moins un dépôt; un éditeur approvisionne au moins un dépôt; tout dépôt contient au moins un livre d'au moins un éditeur.
  Les cardinalités de "LIVRE - STOCKER" sont (1,n).
  Les cardinalités de "EDITEUR - STOCKER" sont (1,n).
  Les cardinalités de "DEPOT - STOCKER" sont (1,n).

7. La représentation graphique

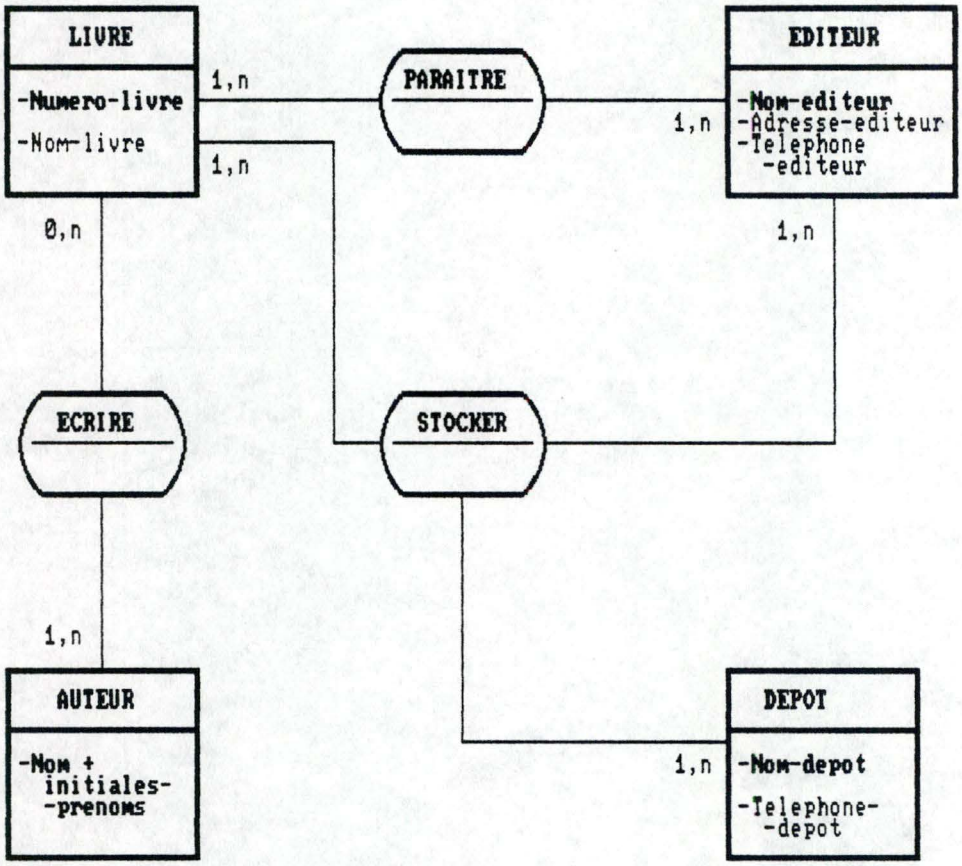


ENTITE



ASSOCIATION

8. Le schéma du système de distribution de livres



9. Simplification du schéma

Une simplification du modèle initiale ne serait pas vraiment bénéfique. Nous laisserons donc le schéma comme il est.

Annexe D

La Normalisation

L'exemple de normalisation repris ci-dessous est fortement inspiré de [HAI 1]. Il se limite aux trois premières formes normales d'une relation.

Soit une relation CO (NCOM, CLI, Adresse CLI, Ligne).

- Si le n-uplet (n,c,a,l) appartient à CO, alors la commande (n) du client (c) dont l'adresse est (a), contient les lignes (l).

- Si (p,q) appartient à Ligne, alors il décrit une ligne spécifiant (q) unités du produit (p).

Les hypothèses de base sont qu'un client n'a qu'une adresse, et qu'un commande ne spécifie au plus qu'une fois le même produit.

La relation pourrait être représentée comme suit :

CØ	NCOM	CLI	Adresse CLI	LIGNE	
				Produit	Quantite
	217	c297	rue de Namur, Bouge	p14	10
				p82	200
	86	c124	avenue Louise, Bruxelles	p51	45
				p12	22
	113	c297	rue de Namur, Bouge	p57	1

Le passage en 1FN de CO suppose l'élimination de l'attribut répétitif Ligne. On obtient alors la relation C1 dont l'extension est présentée ci-dessous :

C1	NCOM	CLI	Adresse CLI	Produit	Quantite
	217	c297	rue de Namur, Bouge	p14	10
	217	c297	rue de Namur, Bouge	p82	200
	86	c124	avenue Louise, Bruxelles	p51	45
	86	c124	avenue Louise, Bruxelles	p12	22
	113	c297	rue de Namur, Bouge	p57	1

Dans C1, pour chaque produit donné, on retrouve la référence du client qui l'a commandé ainsi que son adresse. Cette redondance est éliminée en passant en 2FN, en éclatant C1 en deux relations C11 et C12 :

- C11 (NCOM, CLI, Adresse CLI)

- C12 (NCOM, Produit, Quantité)

dont les extensions sont reprises ci-dessous.

C11	NCOM	Produit	Quantite
	217	p14	10
	217	p82	200
	86	p51	45
	86	p12	22
	113	p57	1

C12	NCOM	CLI	Adresse CLI
	217	c297	rue de Namur, Bouge
	86	c124	avenue Louise, Bruxelles
	113	c297	rue de Nanur, Bouge

Si C11 ne contient plus d'anomalies, il n'en est pas de même pour C12 dans laquelle l'adresse d'un client apparaît chaque fois que celui-ci a passé une commande. Pour passer en 3FN, C12 est éclatée en deux relations distinctes C121 et C122 :

- C121 (NCOM, CLI)

- C122 (CLI, Adresse CLI)

C121	NCOM	CLI
	217	c297
	86	c124
	113	c297

C122	CLI	Adresse CLI
	c297	rue de Namur, Bouge
	c124	avenue Louise, Bruxelles

La relation initiale C0 a donc laissé la place à trois relations C11, C121 et C122 éliminant toute redondance.

Le petit exemple que nous venons de traiter illustre le phénomène d'éclatement des relations.

Si l'on considère que chacune des relations constitue un fichier, on constate que, pour C0, l'ouverture d'un seul fichier permet d'accéder à toutes les informations, alors qu'après normalisation, trois fichiers (C11, C121, C122) doivent être simultanément ouverts pour obtenir les mêmes informations. Par ailleurs, des comparaisons inter-fichiers seront incontournables pour reconstituer la même information. Par exemple si l'on désire connaître les produits commandés par un client, un parcours unique dans C0 sera utile. Pour obtenir la même information après normalisation, la consultation simultanée de C11 et de C121 sera nécessaire. En cas de fichiers séquentiels triés, il faudra parcourir en moyenne la moitié de C11 par client de C121. En cas de S.G.B.D. relationnel, celui-ci réalisera lui-même cette opération (appelée jointure). Néanmoins, les performances seront nettement moins bonnes que dans le cas de C0.

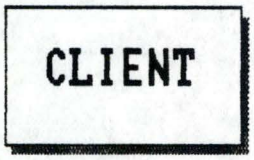
En ce qui concerne l'espace mémoire requis, on peut estimer que le passage de C0 à C11, C121 et C122 a permis de gagner à peu près 30% de place, ce qui n'est pas néligeable.

Si l'on considère donc ces deux critères, il en résulte que le concepteur de la base de données devra tenir compte des accès aux données pour construire sa base le plus efficacement possible.

Annexe E

Le Diagramme des Flux de Données

L'acteur externe :



Le traitement :

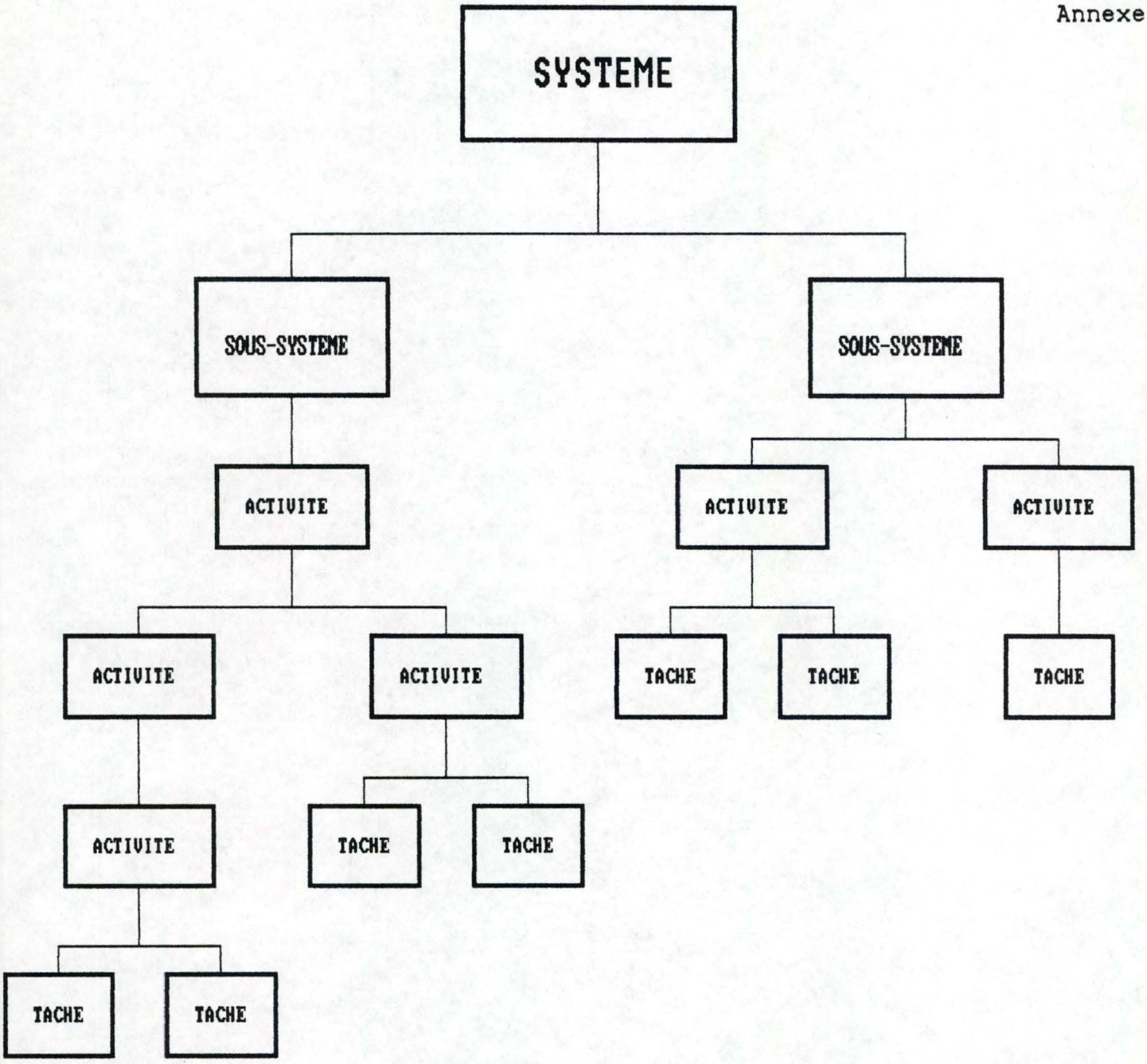
Le flux :

Le stockage :

Annexe F

La Structuration d'un système

Annexe F

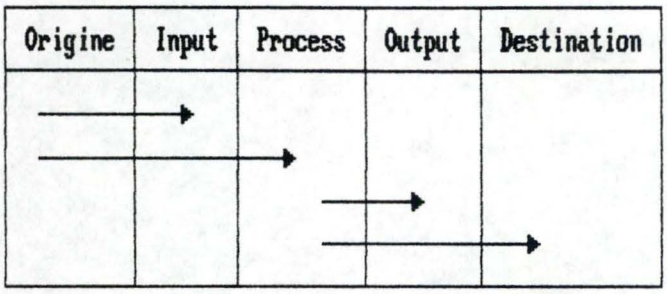


La Structure Hiérarchique des Activités et Tâches.

Annexe G

Le Diagramme I.P.O.

Représentation générale d'un I.P.O. complet :



Exemple de spécification d'un traitement à un niveau conceptuel via un I.P.O. "simple". L'entrée est un bon de commande valide, le traitement consiste à enregistrer ce bon, et la sortie est un message indiquant que le bon a été enregistré. De plus, la cohérence de la base de données garantie en entrée est conservée.

Input	Process	Output
-Bon de Commande valide -BD coherente	Enregistrement d'une commande	-Commande enregistree -BD coherente

Le diagramme I.P.O. ci-dessous représente la spécification fonctionnelle d'un traitement élémentaire. L'entrée est une ligne d'un bon de commande, son origine est le bon de commande, la sortie est un message indiquant que soit la ligne a été mémorisée, soit elle a été refusée. Sa destination est la mémoire du système. Par soucis de clarté, nous avons volontairement omis de spécifier le traitement.

Origine	Input	Process	Output	Destination
-Bon de commande valide	-Ligne de Bon de Commande -BD coherente	Verification d'une ligne de commande	-Ligne memorisee ou Ligne refusee -BD coherente	-Memoire du systeme

Annexe H

Construction d'un algorithme en Langage Structuré

Spécifications du problème :

- soient A et B deux nombres entiers;
- si A positif alors donner à S la valeur du produit de A par B, sinon donner à S la valeur 0.
- les opérateurs disponibles sont l'addition et la soustraction.

La représentation de l'algorithme en pseudo-code est donnée ci-dessous :

```
S = 0
LOOP  WHILE  (A > 0)
      S = S + B
      A = A - 1
ENDLOOP
```

Un exemple un peu plus complexe déterminant le tarif de transport de colis est donné ci-dessous :

```
IF   AVION
     THEN SELECT WHEN Poids < 2kg
                      Prix = 60 francs
                 WHEN 2kg <= Poids <= 20kg
                      Prix = 30 francs/kg
                 WHEN Poids > 20kg
                     Prix = 600 francs
                            + 20 francs/kg
     ENDSELECT
ELSE IF  (pas Belgique) ET express
         ET  (Poids <= 20kg)
         THEN Prix = 30 francs/kg
     ELSE Prix = 20 francs/kg
     ENDIF
ENDIF
```

Annexe I

Les Tables de Décision

La table de décision représentée ci-dessous est une table à entrée limitée. Chaque condition (C1 -> C4) ne peut prendre que deux valeurs : Oui ou Non. Le nombre total de règles est donc de 2**4 = 16. Les actions sont A1 et A2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C1	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
C2	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
C3	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
C4	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
A1	X	X	X	X	X	X		X	X	X	X	X	X		X	
A2				X			X							X		X

La table reprise ci-dessous est une table à entrée mixte. Les conditions C1 et C2 sont binaires, mais C3 peut prendre trois valeurs. Le nombre total de règles est de 2*2*3 = 12. Les actions sont A1 -> A4.

C1	<= 6 mois						> 6 mois					
C2	Y	Y	Y	N	N	N	Y	Y	Y	N	N	N
C3	1	2	3	1	2	3	1	2	3	1	2	3
A1	-	-	-	-	-	-	-	-	X	-	-	-
A2	-	-	-	-	-	-	-	X	-	-	-	-
A3	-	-	-	-	-	-	X	-	-	-	-	-
A4	-	-	-	-	-	-	-	-	-	X	X	X

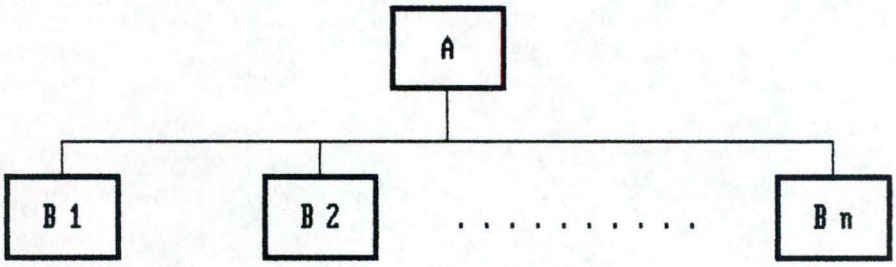
Annexe J

Les éléments de base de J.S.P.

La représentation des programmes

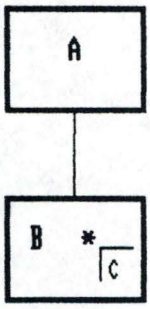
a: Composant-Séquence

Un composant-séquence représente un morceau de programme dont l'exécution correspond à l'exécution successive d'un ou plusieurs autres morceaux de programme. La représentation d'un composant séquence est reprise ci-dessous.



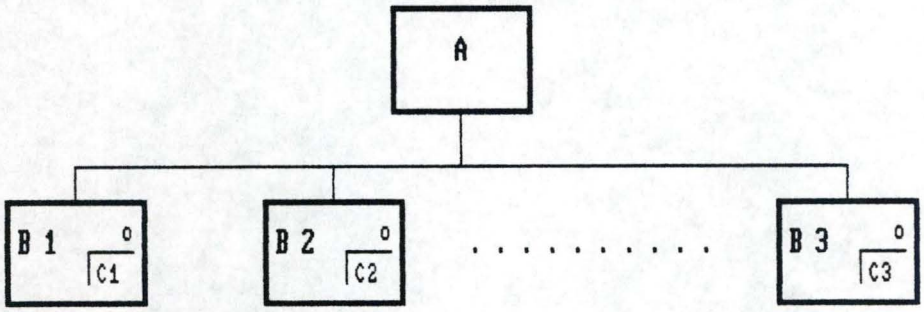
b: Composant-Itération

Un composant-itération représente un morceau de programme dont l'exécution correspond à 0, une, ou plusieurs exécutions d'un même morceau de programme. On exécute B tant que C est vrai. L'astérisque indique la possibilité d'itération.



c: Composant-Sélection

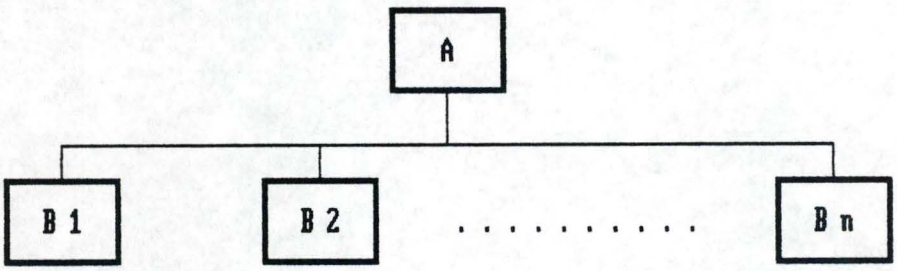
Un composant-sélection représente un morceau de programme dont l'exécution correspond à l'exécution d'un morceau de programme, choisi parmis plusieurs. On exécute Bi si Ci. Le cercle dans le coin supérieur droit indique l'exclusion mutuelle des composants Bi.



<u>La représentation des données</u>

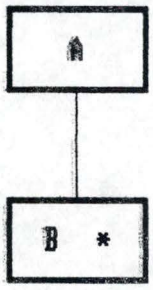
a: Composant-Séquence

Un composant-séquence représente une classe de flux de données obtenus par concaténation de flux de données appartenant à plusieurs autres classes.



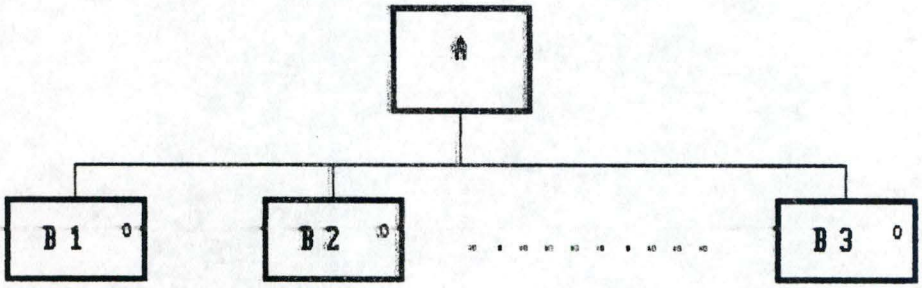
b: Composant-Itération

Un composant-itération représente une classe de flux de données obtenus par concaténation de 0, un ou plusieurs flux de données appartenant à une même classe.



c: Composant-Sélection

Un composant-sélection représente une classe de flux de données égale à la réunion d"un ensemble d'autres classes de flux de données.

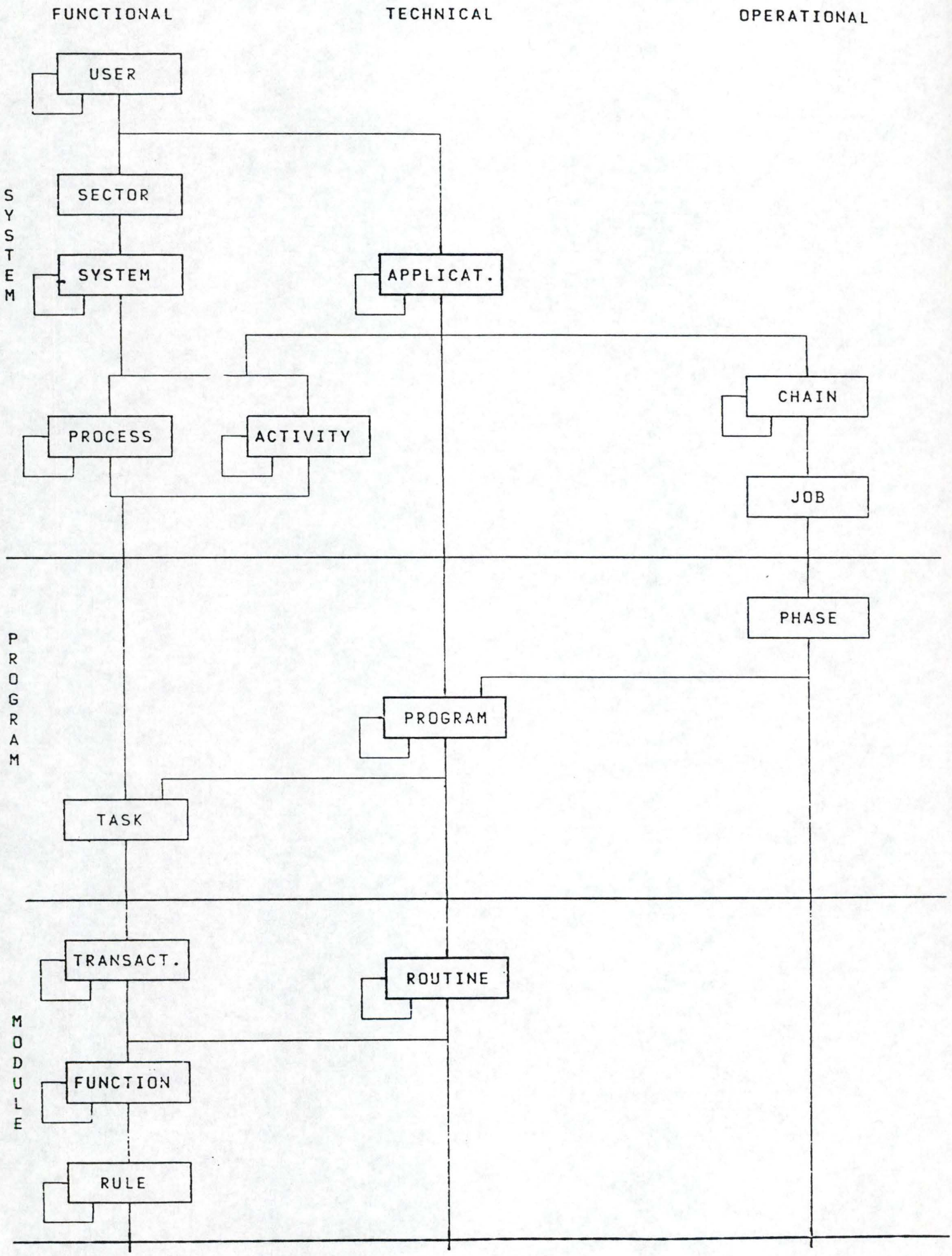


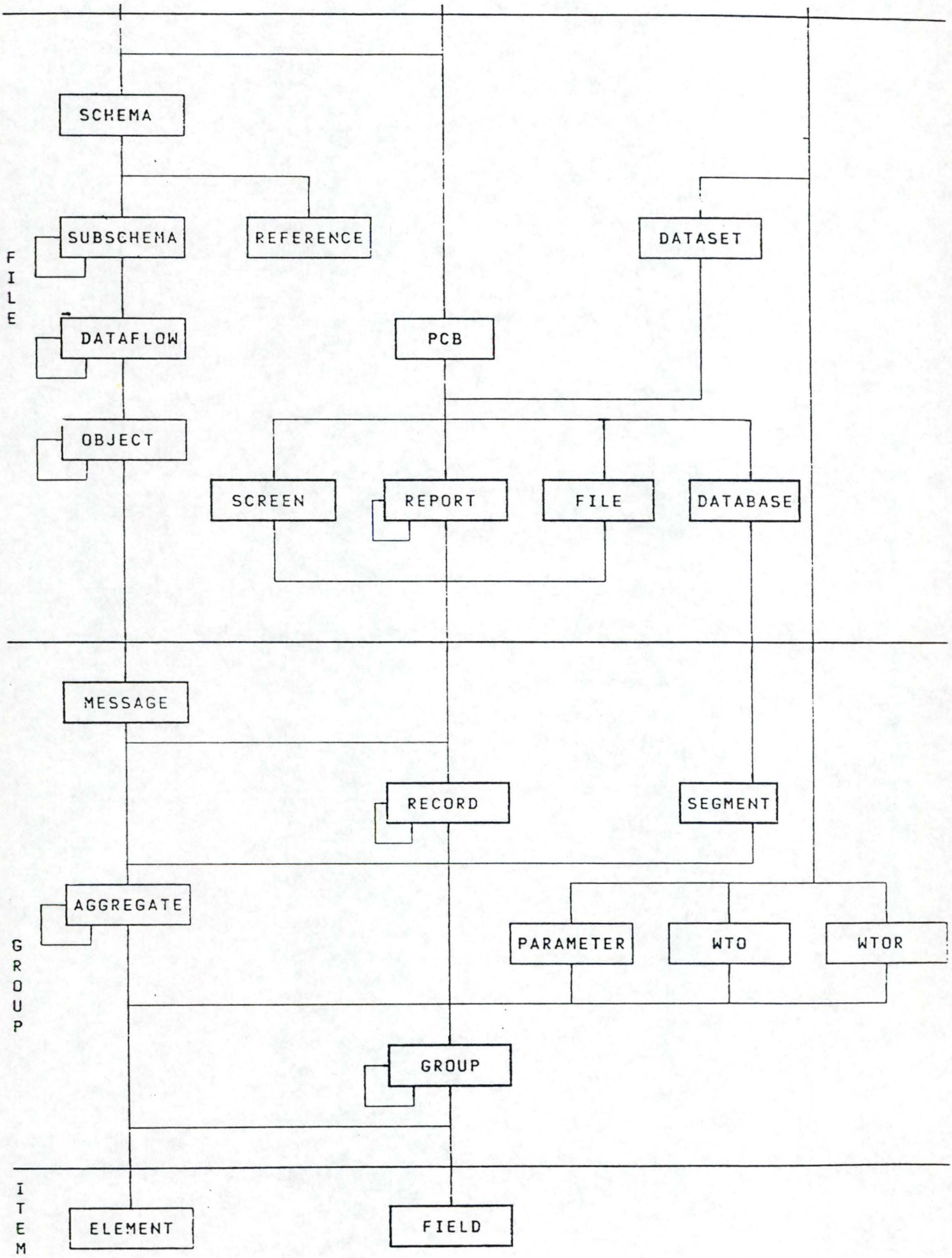
Annexe K

Configuration de DataManager à la C.G.E.R.

Annexe K

DATAMANAGER : HIERARCHY OF MEMBER TYPE



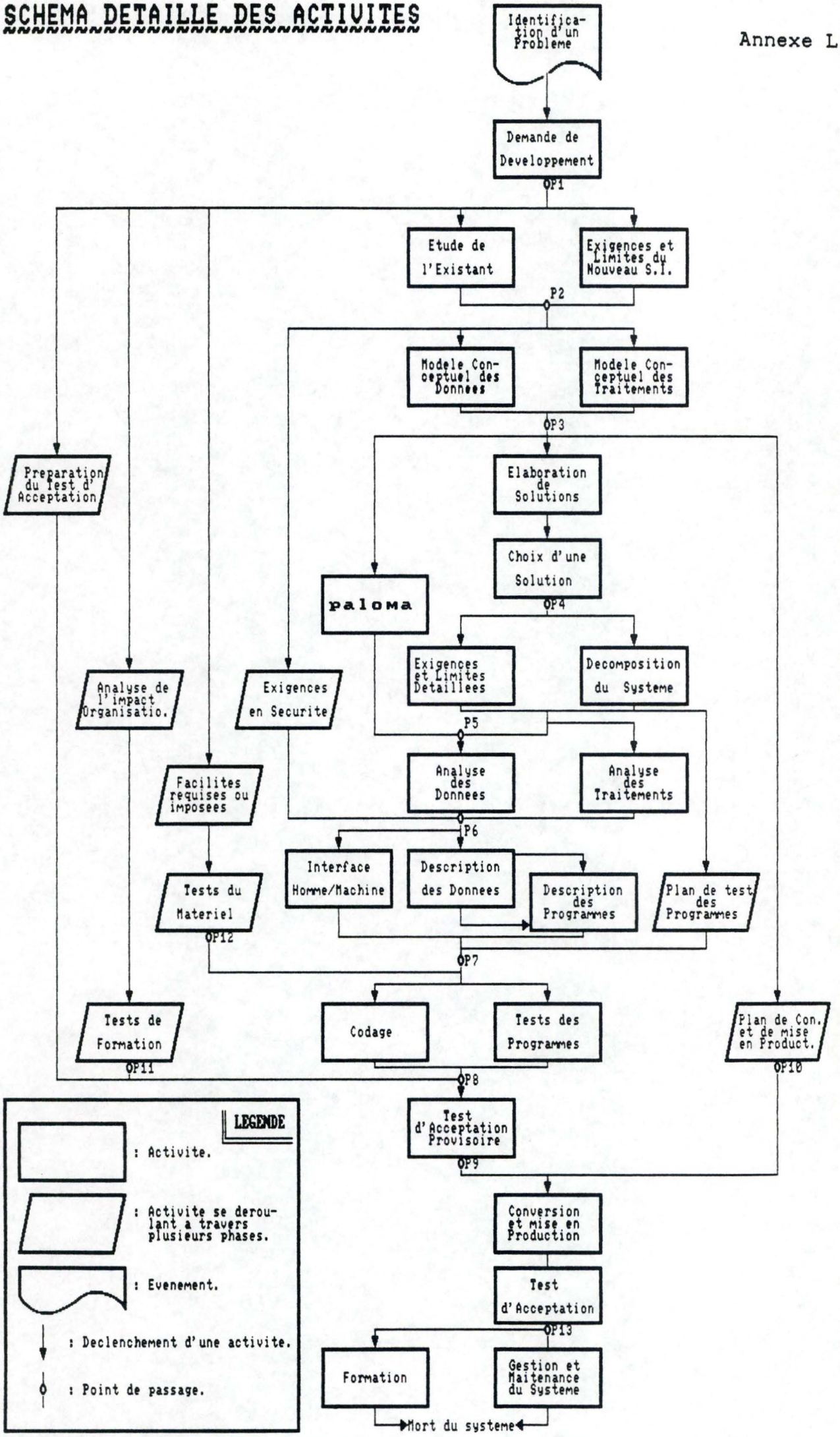

SCHEMA
SUBSCHEMA
REFERENCE
DATASET
FILE
DATAFLOW
PCB
OBJECT
SCREEN
REPORT
FILE
DATABASE
MESSAGE
RECORD
SEGMENT
AGGREGATE
PARAMETER
WTO
WTOR
GROUP
GROUP
ITEM
ELEMENT
FIELD

Annexe L

Schéma détaillé des activités

SCHEMA DETAILLE DES ACTIVITES

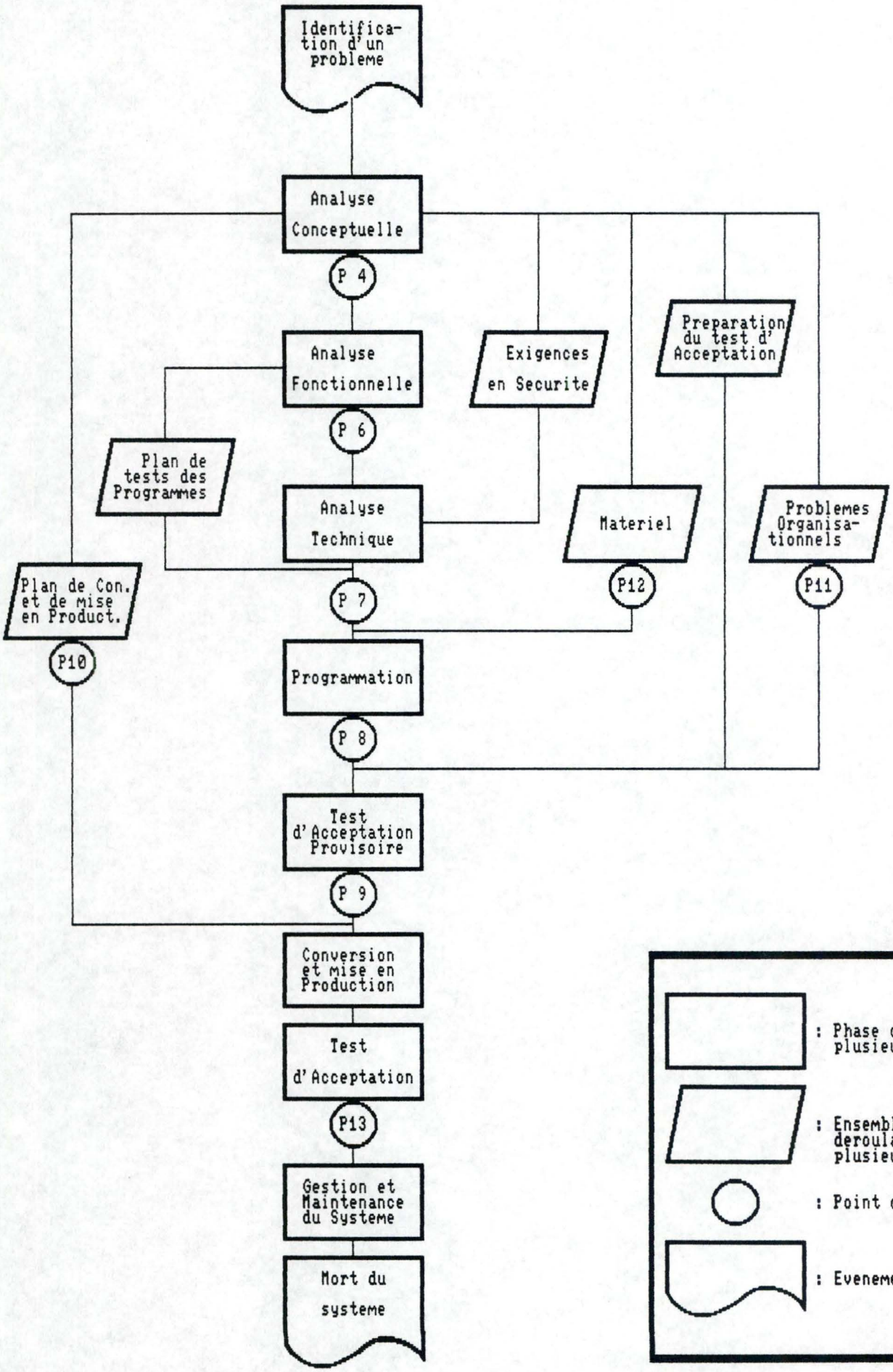
Annexe L

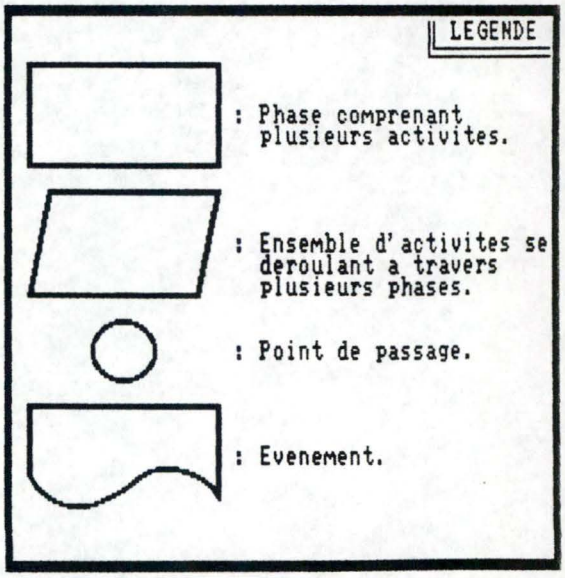


Annexe M

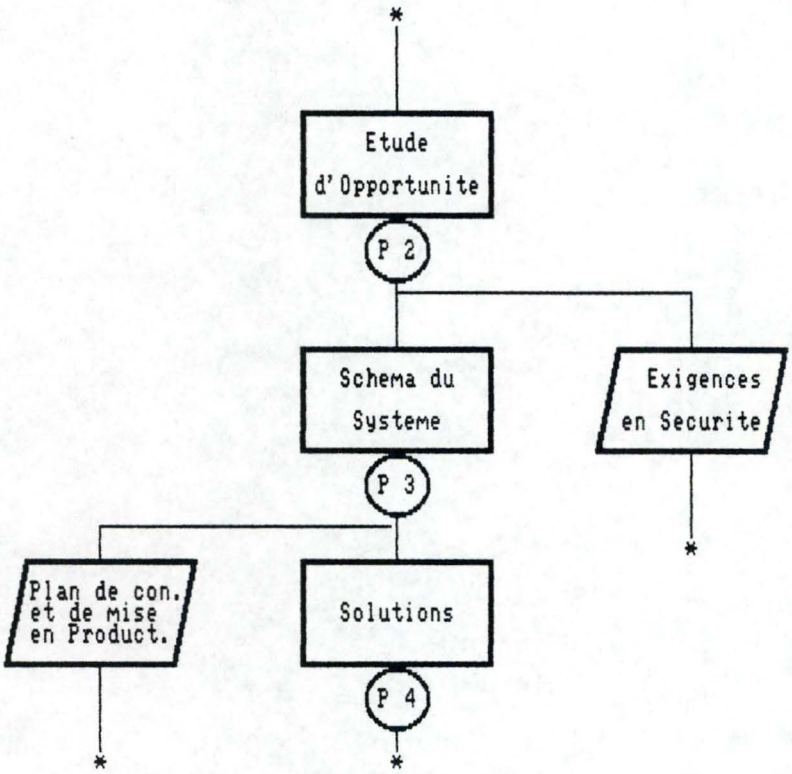
Schéma par raffinements successifs des activités

ACTIVITES A EXECUTER A TRAVERS LES DIFFERENTES PHASES

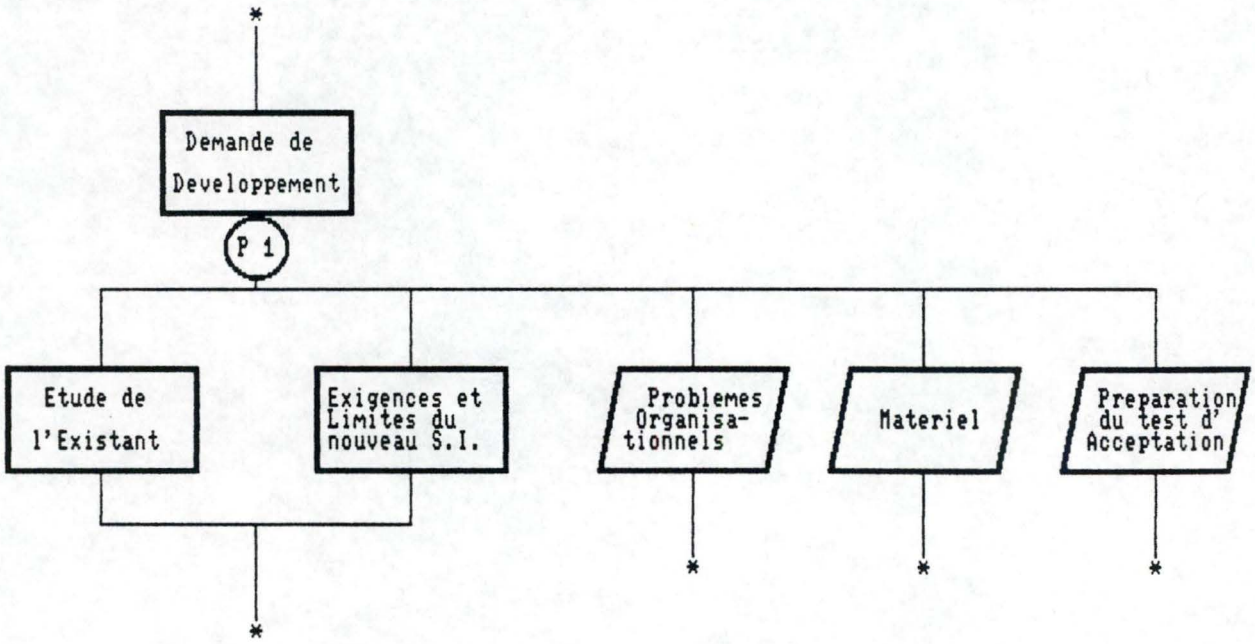




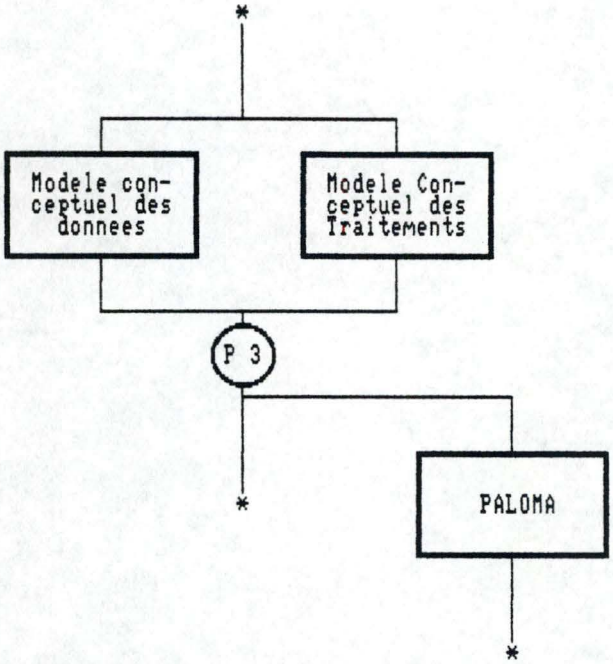
DESCRIPTION DE L'ANALYSE CONCEPTUELLE



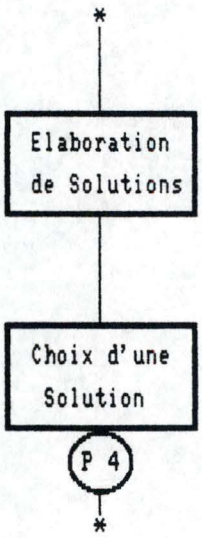
Etude d'Opportunite



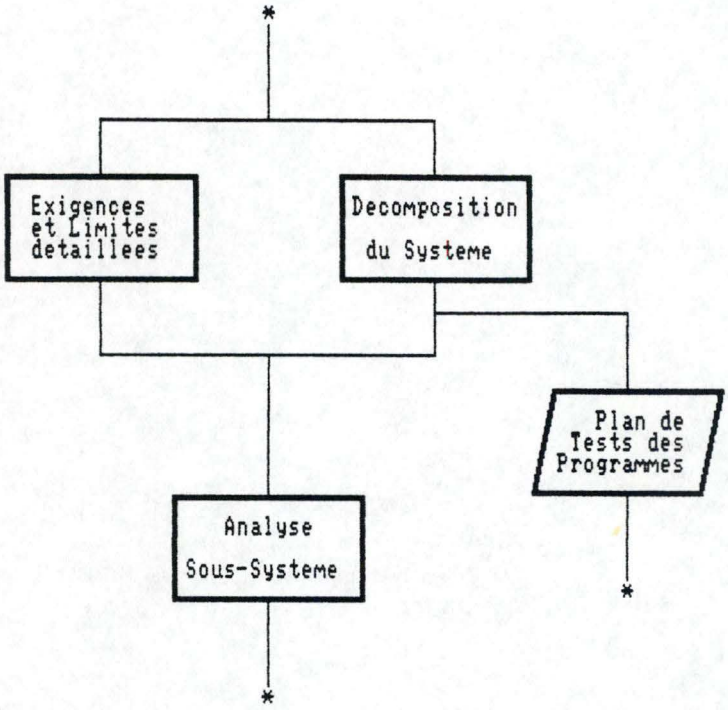
Schema du Systeme



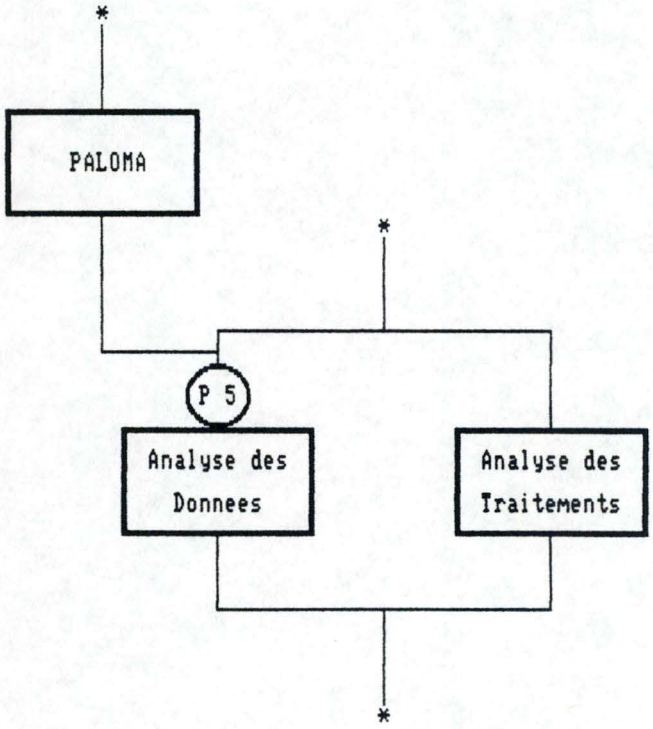
Solutions



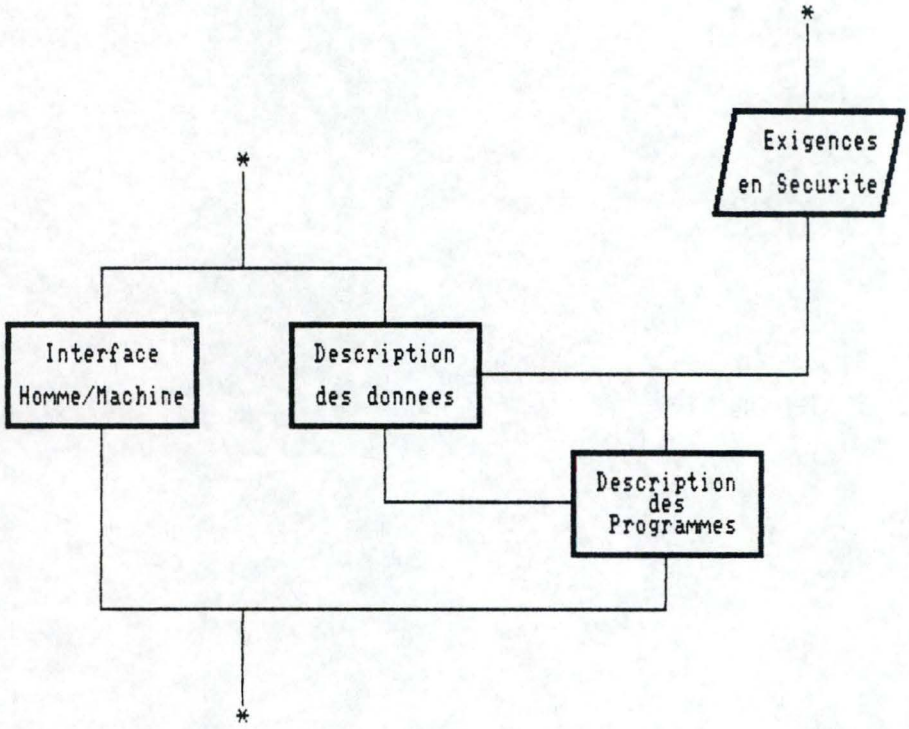
DESCRIPTION DE L'ANALYSE FONCTIONNELLE



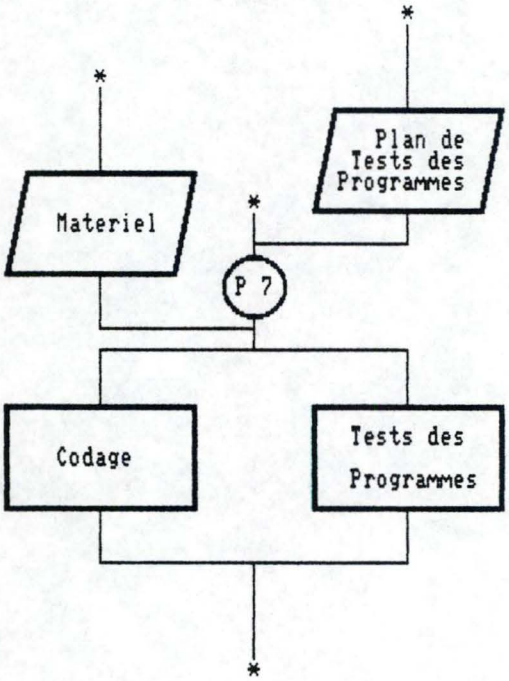
Analyse Sous-Systeme



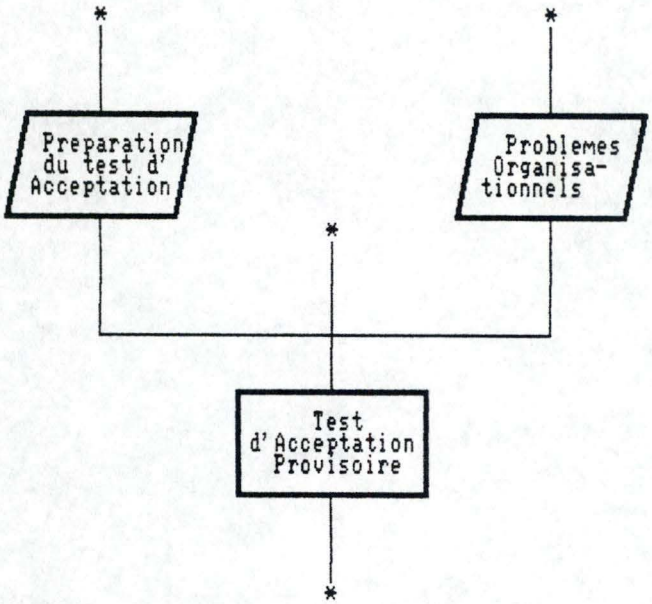
DESCRIPTION DE L'ANALYSE TECHNIQUE



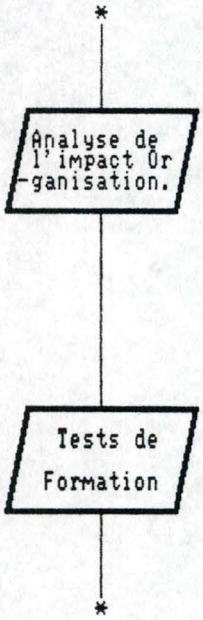
PROGRAMMATION



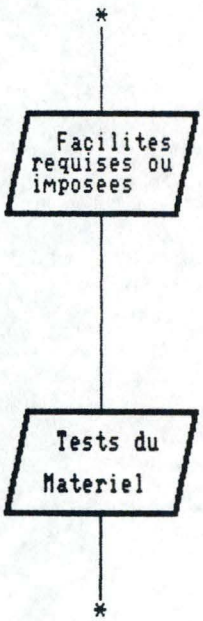
TEST D'ACCEPTATION PROVISOIRE



Problemes Organisationnels



Materiel



Annexe N

Les références aux activités de S.D.M.

Nouvelles activités	Références aux activités SDM
Demande de développement	1.1
Etude de l'existant	1.2 - 1.3
Exigences et limites du S.I.	1.4 - 1.5
Modèle conceptuel des données	1.6
Modèle conceptuel des traitements	1.6
Elaboration de solutions	1.7
Choix d'une solution	1.8
Exigences et limites détaillées	2.1 - 2.2
Décomposition en sous-systèmes	1.6 - 2.3
Analyse des traitements	2.4 - 2.5 - 2.6
Analyse des données	2.9
Exigences en sécurité	2.2 - 2.7 - 3.4
Description des données	3.3
Description des programmes	3.5 - 3.6 - 3.7 - 4.3
Interface Homme - Machine	3.5 - 3.6
Plan de tests des programmes	3.8 - 5.5 - 5.6
Facilités requises ou imposées	1.7.1 - 2.10 - 2.11
Tests du matériel	1.9.2 - 5.2
Codage	4.4 - 4.5
Tests	4.6 - 4.7 - 5.3 - 5.6
Préparation du test d'acceptation	1.9.3 - 5.5
Analyse de l'impact organisa[illegible]	1.9.1-2.2-2.8-3.1-3.2-4.1-4.2
Tests de formation	5.4
Test d'acceptation provisoire	5.7
Plan de conversion et mise en p°.	1.9.1 - 2.13 - 6.1
Conversion et mise en production	6.3 - 6.4 - 6.8
Tests d'acceptation définitif	5.7
Formation	6.2 - 6.5 - 6.7
Gestion et maintenance	phase 7