MASTER EN SCIENCES INFORMATIQUES

Implantation de méthodes de conception et de développement de systèmes d'information. Étude de cas

Marcelis, Didier

Award date:
1990

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix à Namur
Institut d'informatique

Année académique 1989-1990

IMPLANTATION DE METHODES DE CONCEPTION ET DE DEVELOPPEMENT DE SYSTEMES D'INFORMATION. ETUDE DE CAS.

Mémoire de fin d'études présenté par
Didier MARCELIS
en vue de l'obtention du grade de
licencié et maître en informatique

Promoteur: R.P.J. BERLEUR

Résumé

Ce mémoire a pour sujet l'étude de l'implantation de la méthode de développement de logiciels SDM (System Development Method) dans le département informatique de la Caisse Générale d'Epargne et de Retraite. Le processus d'implantation ne sera pas analysé en tant que tel, nous examinerons plutôt quels sont les objectifs d'une telle méthode, les différentes réactions que celle-ci a engendrées, l'influence de l'environnement informatique sur la perception et l'application de SDM par les développeurs, les principaux points forts et points faibles de l'implantation de SDM à la CGER. Notre but n'est donc pas tellement d'analyser la méthode d'un point de vue théorique, mais bien de voir comment celle-ci se comporte dans un environnement de développement particulier.

Abstract

This thesis has for subject the study of the implantation of the method for the design of information systems SDM (System Development Method) in the Caisse Générale d'Epargne et de Retraite. The implantation process will not be analysed for itself, we will rather examine what are the purposes of such a method, the different reactions engendered by this one, the impact of the data processing environment on the perception and the application of SDM by the informaticians, the majors strengths and weaknesses of the SDM implantation in the CGER. Our target isn't an theoretical study of the method but the study of SDM in a particular development environment.

REMERCIEMENTS

Je remercie particulièrement le Père Berleur pour avoir accepté la direction de ce mémoire ainsi que Mme Claire Lobet-Maris pour les précieux conseils qu'elle m'a prodigués.

J'exprime également ma reconnaissance à Marie-Claire Verlaine, chef du service DSIM de la Caisse Générale d'Epargne et de Retraite et à Christian Graas, responsable du groupe méthodes pour leur accueil ainsi que pour le temps qu'ils m'ont consacré.

Je tiens également à remercier toutes les personnes du département informatique de la CGER qui m'ont accordé une entrevue.

Enfin, je remercie toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire.

INTRODUCTION

Dans un monde où l'informatique s'immisce dans la plupart des travaux humains, où elle automatise de nombreux processus manuels; nous pouvons nous demander si elle n'est pas en train d'empiéter également sur son propre terrain. Des outils tels que les générateurs de code, les langages de quatrième génération ou encore les simulateurs de systèmes d'information semblent à la fois faciliter et standardiser les tâches des concepteurs de logiciels et, dans une certaine mesure, ouvrir l'informatique aux non initiés.
Les méthodes de développement de logiciels s'inscrivent dans cette lignée de standardisation de la conception et du développement des systèmes d'information.
Ce mémoire va renter d'étudier différents aspects de l'implantation d'une méthode de développement particulière (la System Development Method) dans le département informatique de la Caisse Générale d'Epargne et de Retraite (CGER).
Nous n'allons pas analyser le processus d'implantation de SDM ayant débuté il y a un peu plus de deux ans, mais plutôt aborder les thèmes suivants:

- quels pourraient être les objectifs d'une méthode de développement de logiciels et quelles sont les raisons pour lesquelles la CGER en a adopté une ?
- Quelles sont les réactions des développeurs face à SDM ?
  Quels sont les éléments qui pourraient expliquer ces réactions ?
- Quelle est l'influence des éléments de l'environnement de développement sur l'application de SDM par les développeurs ?
  Comment expliquer cette influence ?
- Quels sont les points forts et les points faibles de l'implantation de SDM à la CGER ?
  Dans quelle mesure les points faibles peuvent-ils être améliorés ?

La structure suivie sera la suivante:
pour faciliter la suite de la lecture, nous présenterons d'abord succinctement la méthode de travail utilisée pour l'élaboration du mémoire, la Caisse Générale d'Epargne et de Retraite et son environnement, les principaux outils et modèles d'aide aux développements de logiciels présents à la CGER et la méthode SDM.
Le premier chapitre s'intéressera ensuite à l'environnement informatique de la CGER afin de présenter le cadre dans lequel SDM a été instaurée.
Nous examinerons successivement l'environnement du département informatique, la structure et le fonctionnement du département et les moyens de coordination des tâches qu'il utilise.

Le deuxième chapitre présentera ensuite les résultats des interviews menées au sein du département informatique.
Un premier point précisera le cadre général des interviews (grades professionnels, type de tâches des interviewés)
Un deuxième point présentera les réponses obtenues à propos de la méthode SDM (degré d'application de celle-ci par les développeurs, points forts et points faibles de la méthode). Ces résultats seront ensuite interprétés.
Un dernier point tentera de déterminer quels sont les facteurs de l'environnement informatique qui favorisent et quels sont ceux qui défavorisent l'application de la méthode de développement SDM.
Le dernier chapitre présentera quelques propositions d'amélioration sur base des réponses obtenues durant les interviews.
Les propositions concerneront:

- la formalisation des groupes de travail
- l'adaptation de certaines étapes de SDM pour les maintenances de logiciels et pour les adaptations de progiciels
- la circulation de l'information.

1. Méthode de travail

Après avoir pris connaissance de certains points théoriques concernant la structure des organisations, la structure et les stratégies des départements informatiques, nous avons utilisé ces apports théoriques afin de mieux comprendre la structure et le fonctionnement du département informatique de la CGER.
L'enquête menée à la CGER s'est déroulée en deux temps.
Dans un premier temps, nous avons interrogé des "théoriciens" de SDM: un chef de service et un méthodologue initiateurs de l'introduction de la méthode à la CGER. Ils nous ont expliqué la structure et le fonctionnement du département ainsi que les raisons et les premiers résultats de l'introduction de SDM.
Dans un deuxième temps, nous avons interviewé une trentaine de membres du département informatique de différents grades hiérarchiques, appartenant à différents groupes de travail et un utilisateur des services informatiques à propos de l'organisation et du contenu de leur travail, des problèmes rencontrés dans leur situation de travail, de l'aide apportée par la méthode, des inconvénients qu'engendre l'application de la méthode SDM,... . Le questionnaire utilisé se trouve en annexe 1.

2. Présentation succincte de la CGER

Créée en 1865 pour collecter l'épargne populaire à l'échelle nationale, la Caisse Générale d'Epargne et de Retraite se trouve dispensée au début de son existence de développer son propre réseau commercial car les bureaux de poste vont lui servir d'intermédiaires habilités à recueillir et à rembourser l'épargne. Cette option initiale va conditionner toute la structure ultérieure de la banque: elle entraîne la concentration et la centralisation administrative, et elle isole l'entreprise de son marché. Au cours des années, le réseau des bureaux de poste devient un obstacle à l'expansion après avoir été à la base même de la réussite: pour faire face à la concurrence des banques privées, la CGER n'a d'autre issue que de devenir une organisation polyvalente et compétitive. Le système des bureaux de poste va alors devenir un goulot d'étranglement ne permettant pas de supporter les nouvelles contraintes commerciales de la banque. Un propre réseau d'agences va être constitué à partir de 1959. La politique d'ouverture d'agences atteint son apogée entre 1975 et 1980 en créant jusqu'à 125 agences par an. Cette politique coïncide avec l'informatisation des opérations de guichet.
Le développement du réseau d'agences crée la nécessité d'une décentrali-

sation régionale.
A partir des années 1976-1977, la CGER établit six sièges régionaux regroupés au sein de trois directions (Flandres, Bruxelles et Wallonnie) et qui viennent chapeauter le réseau d'agences. Le poids des économies d'échelle et des rapports de puissance hérités du passé contribua toutefois à limiter les attributions accordées aux sièges régionaux. Les équipements et les développements informatiques par exemple sont presque totalement sous contrôle du siège central.
La CGER ne mérite plus aujourd'hui son nom. L'épargne traditionnelle lui échappe peu à peu et son rôle dans les retraites s'ammenuise. La CGER est effectivement devenue durant ces dernières années une banque semblable aux grandes banques privées de notre pays.
Quatrième banque belge, 1100 agences environ, un peu plus de 10.000 employés, la CGER reste une banque publique. En 1980, le législateur l'a autorisé à se déspécialiser au point de pouvoir fournir les mêmes services que les banques privées. Le législateur a imposé dans le même mouvement une division comptable ainsi que la création de deux conseils d'administration et de deux directions distinctes pour les activités de banque et pour les activités d'assurance. Cette division se retrouve, comme nous le verrons plus loin, au sein de la structure du département informatique.
La CGER a donc complété l'éventail de ses instruments: création de sociétés de leasing et de factoring, l'assurance-auto, les filiales à l'étranger,... .
Cet élargissement d'activités a eu un effet considérable sur l'informatique puisque celle-ci était obligée de suivre le mouvement. Les besoins des utilisateurs du département sont devenus plus nombreux, leurs demandes toujours plus complexes et plus pressantes. De plus, le département est, comme pour la plupart des départements informatiques des grandes entreprises, immergé dans la maintenance: il doit soutenir et adapter l'existant en place. La maintenance est devenue très lourde. D'après un informaticien: " *Ce qu'on a mis au point est d'une telle complexité que tenter d'intervenir là-dedans c'est courir au suicide*" . Bien que la bibliothèque des programmes contient environ 10000 logiciels, de nombreuses demandes économiquement justifiées doivent parfois attendre des années avant de se voir octroyer un projet informatique. C'est dans ce cadre qu'est venue s'instaurer la méthode SDM.

3. Présentation des principaux outils et modèles d'aide au développement de logiciels

Afin de faciliter la lecture de la suite, nous allons présenter, sans entrer dans les détails, les principaux outils et modèles qui, pour la plupart, existaient à la CGER avant l'introduction de SDM et qui sont préconisés dans le cadre de la méthode.

3.1. Les modèles d'aide au développement

3.1.1. La modélisation conceptuelle des données

La méthode retenue est celle du schéma Entité-Relation.
Ce modèle aide à identifier les informations nécessaires et à établir des liens entre ces informations.
Le modèle est décrit en détail par F. BODART et Y. PIGNEUR (1989, pp. 12-49).
De cette description retenons que le schéma s'obtient en suivant les étapes suivantes:

- élaboration de la liste brute des informations. Toutes les informations que va utiliser le futur S.I. y seront regroupées. L'élaboration de cette liste commencera lors de l'étude de l'existant.
- épuration de la liste brute. Les synonymes, les redondances seront éliminés lors de cette étape. L'épuration fournira une liste des propriétés des objets
- dégagement des types d'entités
- rattachement des différentes propriétés aux entités
- définition des relations inter-entités
- détermination des cardinalités
- simplification éventuelle du modèle (contraintes d'intégrité)
- quantification du modèle conceptuel (nombre d'occurences estimé).

3.1.2. Le modèle dynamique des données

Le modèle dynamique des données de la CGER résulte de l'exécution de PALOMA (voir infra point 3.2.2). Il indique les points d'entrée dans le modèle conceptuel et estime le trafic inter-objets. Les Data Base Administrators (DBA) s'appuient sur ces estimations pour la transformation du modèle conceptuel en modèle logique des données.

3.1.3. Le modèle logique des données

Le modèle d'accès généralisé (MAG) "*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* ". "*Le MAG sera mis au point sur base du modèle conceptuel des données et éventuellement lors de la normalisation* " (HAINAUT, 1988, p.20). Des calculs d'estimation de trafic et d'accès aux données précis seront d'un grand secours pour l'élaboration d'un MAG performant

3.1.4. Le diagramme de flux de données

Un diagramme de flux de données (DFD) est un graphe des flux d'information circulant dans une organisation ou dans une partie de celle-ci.
A la CGER, les DFD sont composés de quatre éléments de base:

- les FLUX représentent l'information circulant dans l'entreprise
- les TRAITEMENTS décrivent les transformations opérées sur les informations
- les STOCKAGES reprennent les informations stockées quelque part
- les ACTEURS EXTERNES sont les sources ou les destinations externes des flux.

Un diagramme de flux de données global sera successivement décomposé en diagrammes plus détaillés pour arriver en bout de course à des DFD dont les traitements sont élémentaires (tâches).

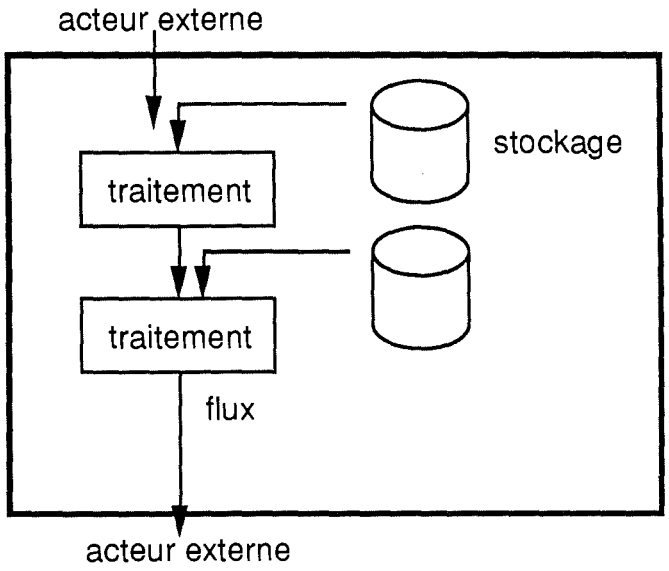


figure 1.1: exemple de DFD simplifié.

3.1.5. La structure hiérarchique des activités et tâches (SHAT)

La SHAT est une technique permettant de décomposer un système en sous-systèmes, les sous-systèmes en activités et les activités en tâches. Le problème initial est décomposé en sous-problèmes plus simples et ainsi de suite. Le résultat est une arborescence dont les différents éléments sont caractérisés par leur niveau de détail. Il s'agit donc d'une découpe verticale des différents niveaux de DFD's.

3.1.6. Le diagramme hierarchical input process output (HIPO)

Le diagramme HIPO est une simplification du modèle de la statique des traitements proposé par BODART et PIGNEUR (1989, pp. 63-65).
Le diagramme HIPO décrit précisément chaque tâche, il se présente sous la forme d'un tableau comportant cinq colonnes:

- la première identifie l'origine de chaque élément (information) en entrée
- la deuxième décrit les inputs nécessaires à l'exécution de la tâche
- la troisième décrit le traitement effectué sur les inputs pour obtenir les outputs. La description peut se faire en pseudo-code (voir le point 3.1.7), en JSP (voir point 3.1.8), grâce aux tables de décision, aux automates déterministes ou aux formules mathématiques.
- la quatrième décrit les outputs de la tâche
- la cinquième indique la destination de chaque élément en sortie.

3.1.7 Le pseudo-code

Le pseudo-code est un langage pourvu d'une syntaxe réduite et d'un vocabulaire précis.
Il va permettre de décrire précisément les traitements d'un système d'information. La programmation consistera à traduire ce pseudo-code en un langage de programmation.
Le pseudo-code utilise les trois structures de base que sont la séquence, l'itération et la sélection.

Exemple simplifié de pseudo-code:

```
lire CLIENTS
LOOP
while not fin-fichier
        if solde (:client) < 0
        then...
        endif
        lire CLIENTS
ENDLOOP
```

3.1.8. Jackson Structured Programming (JSP)

Cette méthode est principalement destinée à analyser et à décrire les programmes.
Son principe de base est la description des flux séquentiels du problème et leur mise en correspondance.
Les données en entée et en sortie sont présentées sous forme arborescente à l'aide d'itérations, de sélections et de séquences.
Les deux arborescences sont ensuite analysées de manière descendante afin de découvrir les parallélismes sur base de la décomposition établie entre les entrées et les sorties.

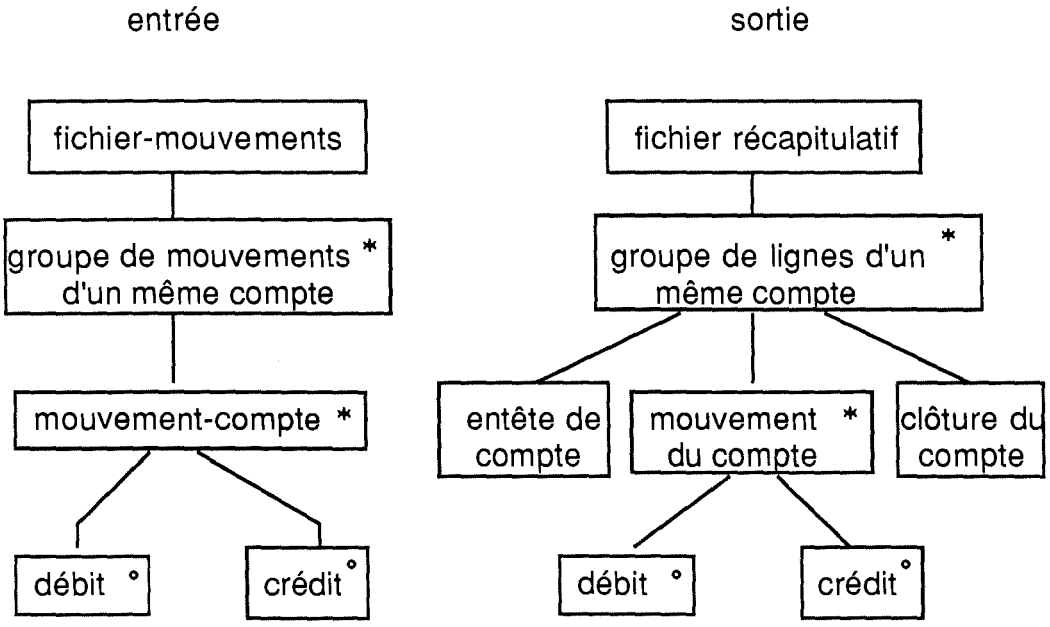


figure 1.2: exemple d'algorithme produisant des extraits de compte

3.2. Les outils d'aide au développement

3.2.1. Le dictionnaire de données DATAMANAGER

Un dictionnaire de données est un système d'information sur les données et les concepts utilisés lors des développements de logiciels.
Le dictionnaire de données DATAMANAGER (DMR) est utilisé à la CGER depuis plusieurs années.
Il centralise la documentation relative aux systèmes d'information. Ainsi, tous les concepteurs ont à leur disposition la définition de chaque concept qu'ils auront à utiliser.
DMR reconnaît six types d'entités ou types de membres de base:

- les FILE, GROUP et ITEM pour les données
- les SYSTEM, PROGRAM et MODULE pour les traitements.

Sur base de ces types de base, une quarantaine de types de membres ont été décrits. Ils se groupent en membres de type "FUDOS" (analyse conceptuelle et fonctionnelle) "TEDOS" (analyse technique et programmation) et "PRODOS" (production).
Lors de son introduction dans le dictionnaire, chaque membre devra être décrit à l'aide de clauses descriptives et de clauses relationnelles (relations entre les membres de type de membre) en fonction de son type.
L'utilisateur pourra accéder aux informations contenues dans DMR à l'aide d'un langage de manipulation de données (modification de certaines données et interrogation des données).

3.2.2. PALOMA

Le logiciel PALOMA (PAth LOad MAtrix) a été développé à LA CGER.
PALOMA estime la charge supportée par chaque arc reliant les diverses entités du modèle conceptuel des données.
PALOMA se base sur les informations introduites dans DMR et il sert notamment à la validation du modèle conceptuel des données.
L'étape PALOMA est un point de passage obligé pour tous les développements.
Les résultats fournis par le logiciel vont guider les Data Base Administrators pour la transformation du schéma conceptuel en un schéma logique des données. Une description plus précise de ce logiciel est faite au point 3.3.2 du troisième chapitre.

4. Présentation de la méthode SDM

Le groupe méthodes (1989, p.1) reprend les raisons pour lesquelles SDM peut être employée:

- *elle propose une checklist pour le développement et la gestion d'un système d'information*
- *elle est une base pour le planning et le contrôle de l'avancement et de la qualité des projets*
- *elle constitue un moyen d'uniformisation*
- *elle est un outil de communication entre les utilisateurs et les spécialistes informatiques*
- *elle sert de base pour la documentation d'un projet*

SDM est une méthode théorique globale qui doit être adaptée à l'environnement dans lequel elle sera appliquée. A cet égard, un résumé de la méthode a été élaboré, il reprend pour chaque étape de développement: son nom, le responsable à contacter, le numéro de téléphone de ce responsable et les techniques applicables a été élaboré à cet égard.
Ce résumé est repris en annexe 2.

SDM comprend sept phases:

- l'étude de l'existant et l'analyse conceptuelle
- l'analyse fonctionnelle
- l'analyse technique
- la programmation
- les tests
- la conversion et la mise en production (conversion des données et préparation de l'organisation)
- la gestion et la maintenance du système.

SDM décrit donc étape par étape les phases d'un projet de développement de logiciels.
Ces étapes sont exécutées selon un ordre précis (voir à ce propos l'annexe 3).
Il se peut toutefois que dans la pratique l'application de certaines étapes devient quasi impossible ou inutile.
Prenons quelques exemples:

- une bonne documentation existe déjà (certaines étapes ont déjà été partiellement ou totalement exécutées)
- la nature du projet peut rendre certaines étapes inutiles
- certaines activités sont sous-traitées par d'autres départements ou par des tierces personnes
- le laps de temps écoulé entre deux phases peut parfois être fort impor-

tant ce qui implique que des informations recueillies durant la première phase doivent parfois être mises à jour lors de la deuxième phase
- lorsque les exigences ou les buts d'un projet changent en cours de développement , certaines étapes doivent être réeffectuées

...

Nous aurons l'occasion d'approfondir ces points lors des chapitres 2 et 3.
D'autre part, un document volumineux reprenant en détail et pour chaque étape de développement les méthodes applicables pour les réaliser, la manière de travailler, la documentation à établir et les techniques et cours existants à la CGER pour les mener à bien a été terminé en décembre 1989.
Pour illustration, nous avons repris la description complète de l'étape 2.3 de l'analyse fonctionnelle: *diviser le système en sous-systèmes et les décrire* au point 2.6.2 du deuxième chapitre.

Afin de faciliter la lecture de ce qui va suivre, nous reprenons ci-dessous la liste des différentes étapes de SDM.

1. Etude de l'existant et analyse conceptuelle.

1.1 définir le problème et le champ de l'étude
1.2 rassembler les données sur la situation existante
1.3 analyser et évaluer les données
1.4 déterminer les objectifs et exigences du nouveau système
1.5 arrêter les points qui restent à résoudre et les hypothèses de base
1.6 faire un schéma du système
1.7 déterminer les outils et les solutions possibles
1.8 évaluer les solutions et opérer une sélection
1.9 déterminer les problèmes de conversion et de mise en production et déterminer les critères d'acceptation
1.10 élaborer un planning global et une vue d'ensemble des coûts et bénéfices
1.11 rédaction du rapport

2. Analyse fonctionnelle.

2.1 spécifier les exigences du système y compris futures
2.2 déterminer le cadre dans lequel le nouveau système devra fonctionner
2.3 diviser le système en sous-systèmes et les décrire
2.4 définir l'input et l'output par sous-système et les interfaces
2.5/6 élaborer les diagrammes et les descriptions des traitements
2.7 spécifier les exigences en sécurité et en confidentialité
2.8 déterminer les problèmes humains et les solutions

2.9 concevoir la structure logique des données
2.10/12 spécifier les facilités requises en communication de données, en hardware et en software
2.13 élaborer un plan pour la poursuite du développement et la mise en production
2.14 rédaction du rapport.

3. Analyse technique.

3.1 conception des procédures manuelles
3.2 conception des formulaires et de tous les input et output de l'ordinateur
3.3 conception de la structure de stockage
3.4 conception des mesures de sécurité
3.5/6 élaboration des descriptions de programmes et schémas
3.7 description des programmes standards à utiliser
3.8 élaboration d'un plan détaillé de programmation et de tests
3.9 rédaction du rapport

4. Programmation.

4.1 description des tâches
4.2 définir les exigences en personnel et en environnement de travail
4.3 détailler les descriptions de programmes
4.4 codifier les programmes
4.5 compilation et correction
4.6 constituer des données de tests
4.7 tester les programmes
4.8 compléter la documentation des programme.

5. Tests.

5.1 élaboration d'un plan de tests détaillé
5.2 installation du matériel, du software et préparation de l'environnement de travail
5.3 définition des unités de traitement et de leur ordre de succession
5.4 tests des formations, utilitaires et procédures
5.5 constitution des données de test de système et d'acceptation
5.6 tests des sous-systèmes
5.7 exécution du test d'acceptation
5.8 rédaction du rapport "résultat tests"

6. Conversion et mise en production.

6.1 élaborer un plan détaillé de conversion et de mise en production

6.2 formation du personnel informatique
6.3 élaboration d'instructions de conversion et de mise en production
6.4 conversion des données
6.5 donner des renseignements sur le nouveau système
6.6 formation des utilisateurs
6.7 formation du groupe de maintenance
6.8 mise en production du nouveau système et son transfert

7. Gestion et maintenance du système.

7.1 élaboration et utilisation du système de signalement des erreurs
7.2 planification périodique de maintenance
7.3 planification du traitement informatique
7.4 prévention et restauration des fautes et perturbations
7.5 surveillance des dispositions de sécurité
7.6 modification et mise à jour de la documentation. Réponse aux besoins en information ad-hoc
7.7 s'occuper des formations complémentaires
7.8 gestion des données et des fichiers
7.9 évaluation du système et plans d'action

Source Groupe méthodes (1989, pp.1-100)

CHAPITRE 1
L'ENVIRONNEMENT INFORMATIQUE

1.1. Présentation de l'environnement du département informatique.

1.1.1. Les flux d'information à la CGER

Deux types de flux d'information coexistent:

- ceux qui reprennent toutes les relations entre la CGER et ses clients (particuliers, entreprises, institutions)
- ceux qui reprennent toutes les relations internes (comptabilité, personnel, logistique)

A chaque domaine d'intervention de la CGER correspond presque toujours une ou plusieurs applications importantes: assurances, comptes universels, épargne, crédit, bourse, étranger, gestion des bâtiments, administration et gestion du personnel, . . .
Certaines de ces applications s'intègrent partiellement si elles ont des traitements similaires (comptes universels, épargne) par contre, d'autres, ont tendance à se différencier car, de par leur traitement, elles n'ont pas de liens avec d'autres applications (gestion des comptes de l'entreprise par exemple).
L'informatique s'efforce d'intégrer au maximum les applications afin que différents traitements puissent venir s'approvisionner dans la même base de données.
La politique du département était plutôt orientée produit (à chaque produit un logiciel, les différents logiciels n'étant pas interconnectés entre eux). D'un point de vue marketing et stratégique, il est cependant intéressant de développer des applications informatiques orientées client (les différents logiciels ont un point en commun: les clients). Cela permet entre autre choses de connaître tous les produits que possède un client et de déterminer ceux qu'il serait susceptible d'acheter. C'est une des raisons pour lesquelles le département a élaboré le "signalétique client": une application reprenant les principales relations des clients avec la CGER; les divers services peuvent alors utiliser la partie de l'application les concernant. A côté de ce signalétique général gravitent de plus petits signalétiques pour gérer des informations propres à chaque service.

1.1.2. Principales étapes de l'informatisation de la CGER

Pour mieux comprendre les méthodes et les outils de développement utilisés actuellement à la CGER, il nous semble intéressant de tracer un bref historique des principales phases de son informatisation.

Comme nous l'avons déjà explicité plus haut, le fait que la CGER faisait toujours appel à la poste au début des années septante pour la vente de ses produits devint pénalisant. La banque décida pour remédier à cet état de fait de créer son propre réseau d'agences toutes reliées au siège central. Cela impliqua l'usage des télécommunications et, en particulier, la nécessité d'un moniteur de télécommunication capable de gérer tous les transferts d'information. Après un essai avec CICS d'IBM, insatisfaisant du point de vue de la sécurité, la CGER a développé son propre moniteur: le module directeur (MD).
En 1976, les bases de données commencèrent à faire leur apparition et, consciente des avantages quelles pourraient apporter, le département désira implanter IMS d' IBM.
Dans une première phase, il fallait voir si l'entreprise possédait une maturité informatique suffisante pour exploiter correctement les ressources d'IMS. Deux points en sont ressortis:

- les employés avaient le sentiment d'être propriétaires des informations qu'ils manipulaient et n'admettaient pas le partage de leurs fichiers
- les définitions des termes utilisés, l'information, les données n'étaient pas uniformisés. Un travail de synthèse et de modélisation s'imposait car il était impossible d'établir des design IMS à partir d'une base aussi disparate.

En 1978 débutèrent les travaux de modélisation qui concernaient les supports de l'information et le dictionnaire des données.
Vers cette même époque, les directions des entreprises d'une certaine taille commencèrent à s'inquiéter de l'ampleur et de l'évolution spectaculaire des charges informatiques. La CGER ne fit pas exception à la règle et, dans le souci d'améliorer la productivité et la qualité du travail des développeurs, elle commença à s'intéresser aux méthodes de programmation et au niveau de développement conceptuel .
Ce n'est cependant qu'après de nombreuses années que l'apport du niveau conceptuel fut reconnu par tout le monde, lorsque les développeurs et les chefs de projet s'étaient rendu compte qu'une conception préalable permettait une meilleure maîtrise de leurs projets.
L'analyse conceptuelle est devenue aujourd'hui un stade de développement obligatoire.
Le premier essai d'implantation d'IMS fut réalisé pour certains traitements des comptes universels (comptes courants de la CGER). Un premier stade de cette implantation revint à définir toutes les informations que manipulaient les traitements et à définir les règles à appliquer à ces informations.
Le relevé des informations utilisées engendra la création d'un dictionnaire de données manuel qui devint très vite impossible à utiliser vu son ampleur. Un

dictionnaire informatisé fut alors choisi: DATAMANAGER qui est toujours utilisé actuellement. D'un simple support de documentation, il a évolué vers un véritable gestionnaire de données.
A coté de la base de données IMS est venue s'ajouter plus tard DB2.

Nous pouvons classifier les manipulations d'information de la CGER en deux grandes catégories:

- celles qui engendrent de très gros volumes de transactions (par exemple les manipulations portant sur les comptes universels). Ces manipulations sont efficacement servies par IMS.
- celles qui engendrent un volume de transactions plus faible, mais qui font appel à beaucoup d'informations différentes reliées entre elles. DB2 répond très bien à ce type de manipulations.

Les deux systèmes ont donc été jugés souhaitables.
Le moniteur de télécommunication IMS/DC qui avait été implanté permet de faire aussi bien appel à la base de données IMS qu'à la base de données DB2.
Chaque application recevra, en fonction de ses caractéristiques soit un design physique IMS soit un design physique DB2.
Afin d'accélérer la modélisation conceptuelle, le groupe méthodes du DSIM s'est équipé il y a peu de l'outil EXCELERATOR. Celui-ci permet entre autre d'introduire les données de manière graphique (schéma entité/relation,...)
L'explosion de la micro-informatique s'est également fait ressentir à la CGER. La micro-informatique est considérée comme étant à la fois utile et dangereuse. Utile car les PC permettent une certaine décentralisation des développements. Dangereuse car, si au niveau central les règles d'administration de l'information ont été formalisées et sont contrôlées, au niveau de la micro-informatique il est fort difficile d'imposer des règles et des normes: lorsqu'un utilisateur non informaticien développe une petite application grâce aux outils informatiques mis à sa disposition, il ne va en général pas examiner son problème de manière globale et méthodologique. Il ne fait pas d'analyse conceptuelle, il n'envisage pas les éventuels développements futurs, il ne traite pas toujours tous les cas particuliers, il n'envisage pas les interfaces que son application pourrait avoir avec d'autres applications, il ne prend pas en compte les aspects de sécurité et de confidentialité.
Tant que le PC reste isolé, cela ne pose pas de problèmes considérables, ceux-ci surgissent lorsque le PC est connecté à l'ordinateur central ou lorsque plusieurs PC sont connectés entre eux afin d'échanger des informations.
Normalement, le département informatique n'a pas d'autorité sur les utilisateurs, il peut tout au plus les convaincre. Ce n'est que lorsque un PC est connecté à l'ordinateur central que l'utilisateur de celui-ci devra se plier aux

normes imposées par le département informatique chaque fois qu'il désire développer une application.

1.1.3. L'environnement du département informatique

La variation du nombre d'employés dans les différents groupes de développement est, comme nous le verrons plus tard, fort importante. Il s'agit d'un symptôme d'une structure qui ne peut pas planifier ses activités à long terme. Le marché bancaire et le marché des assurances sont fort innovateurs. Lorsque la CGER, tout comme les autres institutions financières, désire lancer de nouveaux produits, elle peut soit mener une politique de leader pour ce nouveau produit ce qui revient à imposer un produit relativement différent de tous ceux qui existent, soit mener une politique de follower ce qui revient à créer rapidement un produit relativement semblable à d'autres.
En cours de vie, un produit bancaire ou un produit d'assurance devra s'adapter en fonction des derniers développements des produits concurrents.
La stratégie du département informatique, à part quelques grandes orientations de fond, est fortement tributaire des projets qu'on lui attribue; ces projets sont directement liés aux priorités des différents produits établies par la direction. La stratégie du département informatique peut être considérée comme un amalgame des stratégies de chaque département de la CGER, elle est par conséquent fort complexe vu la diversité des éléments qui entrent en ligne de compte et vu la vitesse à laquelle chacun des ces éléments évolue.

Dans un environnement stable, une organisation peut prédire les conditions dans lesquelles elle va se trouver à moyen et même à long terme et peut dès lors standardiser ses procédures. Par contre, si une organisation doit faire face à des demandes peu ou pas prévisibles, elle ne peut coordonner ses activités en ayant recours à la standardisation. Elle doit alors utiliser des mécanismes de coordination plus flexibles, moins formalisés: la supervision directe ou l'ajustement mutuel; en d'autres termes, elle doit adopter une structure organique. Un informaticien mentionne: "*nous ne pouvons pas prévoir quelles seront les demandes des utilisateurs dans trois mois*".
D'autre part, si l'on considère la typologie établie par J. WOODWARD (1965), les entreprises peuvent être classées selon leur mode de production:

- la production à l'unité ou en petites séries
- la production de masse
- la production en continu.

La production à l'unité ou en petite série comprend entre autre la production d'unités selon les spécifications établies par les clients. Il semble que les travaux des développeurs de systèmes d'information correspondent bien à cet élément de la typologie de WOODWARD: la demande de développement

ainsi que les spécifications du futur programme sont élaborées en collaboration étroite avec le département demandeur, en fonction du besoin exprimé par cet utilisateur et, sur base de ces spécifications, le département informatique va développer un logiciel unique puisque, à chaque demande de développement, le besoin et donc le logiciel seront nouveaux. Nous sommes donc bien en présence d'une production à l'unité malgré les récupérations d'unités de travail qui se font de manière modulaire.
L' absence de standardisation des produits engendre une absence de standardisation des procédés de fabrication. Les structures du département devront donc être souples et organiques afin qu'elles puissent se modifier en fonction des demandes qui lui parviennent
Le fait que le développement d'un logiciel se fait à l'unité impose un personnel aux compétences techniques très élevées; chaque développement aboutit à un produit unique dont l'élaboration fut elle aussi unique: à partir de l'expérience du personnel et des outils de développement, il faudra "innover" un nouveau produit. D'après un informaticien: "*A chaque demande de développement, il faut créer quelque chose de nouveau*" . Cette affirmation est surtout valable pour l'analyse conceptuelle et l'analyse fonctionnelle, elle l'est beaucoup moins pour la programmation qui se résume en une traduction d'un langage abstrait en un langage compréhensible par l'ordinateur. La programmation pourra de ce fait faire l'objet d'une certaine standardisation. Nous pouvons donc qualifier l'environnement du département informatique de complexe aussi bien à cause de la complexité des produits bancaires/d'assurances que de la complexité des outils et méthodes informatiques à mettre en oeuvre pour élaborer des logiciels uniques et, en général, fort complexes.
H. MINTZBERG (1982, p.258) répertorie quatre types d'organisations sur base de leur environnement:

ENVIRONNEMENT	stable	dynamique
complexe	décentralisée,bureaucratique mécanisme de coordination principal: standardisation des qualifications	décentralisée,organique mécanisme de coordination principal: ajustement mutuel
simple	centralisée,bureaucratique mécanisme de coordination principal:standardisation des procédés de travail	centralisée,organique mécanisme de coordination principal:supervision directe

Figure 1.1: type de structure d'une organisation en fonction de son environnement

En résumé, l'environnement du département informatique de la CGER peut être qualifié de:

- complexe vu:
  - la diversité et la composition des produits financiers/ d'assurances
  - la complexité des techniques de développement de logiciels
  - la complexité des outils de support au développement
- dynamique vu:
  - le caractère unique de chaque demande de développement (production à l'unité)
  - l'évolution rapide des produits bancaires et des produits d'assurances.

H. MINTZBERG souligne que l'environnement a un impact profond sur la structure des organisations, impact souvent plus important que celui de l'âge, de la taille ou du système technique de l'organisation. Les environnements dynamiques paraissent conduire la structure vers un état organique quels que soient par ailleurs son âge, sa taille ou son système technique (1982, p.258).
Cet environnement a eu pour conséquence une augmentation importante des effectifs et des programmes. Le département en serait à l'heure actuelle à plus de 10.000 programmes et à plus de 400 personnes affectées à l'étude, au développement et à l'exploitation des logiciels. Les effectifs ont à peu près doublé en cinq ans ce qui occasionne une certaine crise de croissance dont les effets néfastes seront détaillés lors du deuxième chapitre (difficulté d'obtenir une bonne circulation de l'information, difficulté de faire adopter une méthode de développement de logiciels par tout le monde,...).
Au vu de ce qui est expliqué plus haut, le département informatique ou, tout au moins, le département développements de celui-ci tendrait vers une structure décentralisée et organique dont le principal mécanisme de coordination serait l'ajustement mutuel.
Nous émettons ici l'hypothèse que le département développements du département informatique se rapproche d'une structure adhocratique dans l'acception de H. MINTZBERG: structure décentralisée et organique qui recourt à l'ajustement mutuel pour coordonner ses activités.
Nous allons par la suite être amenés à tester la validité de cette hypothèse.
A coté des caractéristiques principales de l'adhocratie, H. MINTZBERG en a répertorié d'autres:

- partie clé de l'organisation: fonctions de support
- principaux mécanismes de conception: spécialisation horizontale du travail, regroupement des unités sur base des fonctions ou marchés
- facteurs de contingence: environnement complexe, dynamique, système technique sophistiqué et souvent automatisé.

(1982, p.375)

A ce stade, nous avons déjà vu que l'environnement du département est complexe et dynamique.

1.1.4. Limites d'autonomie du département informatique

Un état des réalisations doit être élaboré chaque semestre. Les écarts par rapport aux engagements doivent être justifiés. Le budget accordé chaque année constitue le principal frein à l'autonomie du département: s'il a prévu x employés, le département ne peut en engager plus faute de moyens financiers sauf si une dérogation est accordée par l'administrateur directeur.
Les limites d'autonomie du département informatique sont peu formalisées. Elles ne pourraient être traduites en des règles précises et fixes comme c'est le cas dans les bureaucraties. Les limites d'autonomie se dégagent au cas par cas à la suite d'une négociation entre le département et sa hiérarchie. Si, en cours d'année, le département estime par exemple qu'il lui manque tel matériel ou tel logiciel non prévus au budget, et donc a priori non aquérables, il entamera une discussion avec son administrateur directeur qui peut, si la demande est correctement justifiée, adapter le budget en conséquence. La hiérarchie accepte en général toute proposition fondée, pour autant que les prévisions des Return On Investment soient favorables. Les propositions d'investissement à long terme sont par contre plus critiquées. Comment chiffrer par exemple l'apport d'une méthode de développement de logiciels ou la rentabilité de l'analyse conceptuelle ?

H. MINTZBERG souligne que l'organisation innovatrice ne peut s'appuyer sur aucune forme de standardisation pour coordonner ses activités, elle doit éviter tous les pièges de la structure bureaucratique notamment les comportements très formalisés et l'utilisation intensive des systèmes de planification et de contrôle (1982, p.377).
CHANDLER et SAYLES notent que, comme le travail dans le cadre d'un projet est généralement effectué pour la première fois, les politiques et les précédents sont dans une certaine mesure inappropriés (1971, p.202).

1.1.5. Critères d'établissement du budget

Le budget annuel du département informatique est scindé en trois postes:

- le personnel. Sur base des développements à prendre en charge durant l'année qui suit et repris dans le plan informatique, sont établis le budget personnel statutaire et le budget personnel extérieur (personnel qui est engagé pour faire face aux surcharges temporaires du travail).
- les achats de logiciels. En fonction des besoins du département, des lo-

giciels peuvent être achetés sur le marché. Un certain équilibre s'établit entre le budget achat de logiciels et le budget personnel car, plus les logiciels seront achetés sur le marché, moins les besoins en développement interne seront importants et donc moins importantes seront les dépenses de personnel.

- l'achat et le leasing de matériel informatique et non informatique . Ici également l'attribution du budget est la résultante de la politique définie dans le plan informatique.

Comme nous le verrons plus loin, les processus d'élaboration des plans informatiques et l'établissement des budgets ne sont pas aussi rigides. La hiérarchie reste fort ouverte aux propositions venant du département et, si elle trouve les propositions intéressantes et justifiées, elle débloquera les fonds supplémentaires pour les mettre en oeuvre. D'autre part, des projets supplémentaires prioritaires peuvent survenir suite à l'évolution des marchés financiers; en fonction des besoins, une adaptation rapide des plans et des budgets sera effectuée.

1.2. Structure du département informatique

1.2.1. Présentation de la structure du département informatique de la CGER

La structure et le fonctionnement de la CGER sont nettement subdivisés en deux entités:

l'entité1 regroupe les activités bancaires

l'entité2 regroupe les activités d'assurances

Le département informatique se situe dans l'entité2, il travaille sans distinction pour les deux entités.

Nous vous présentons ci-dessous sa structure (au moment où ce mémoire a été élaboré):

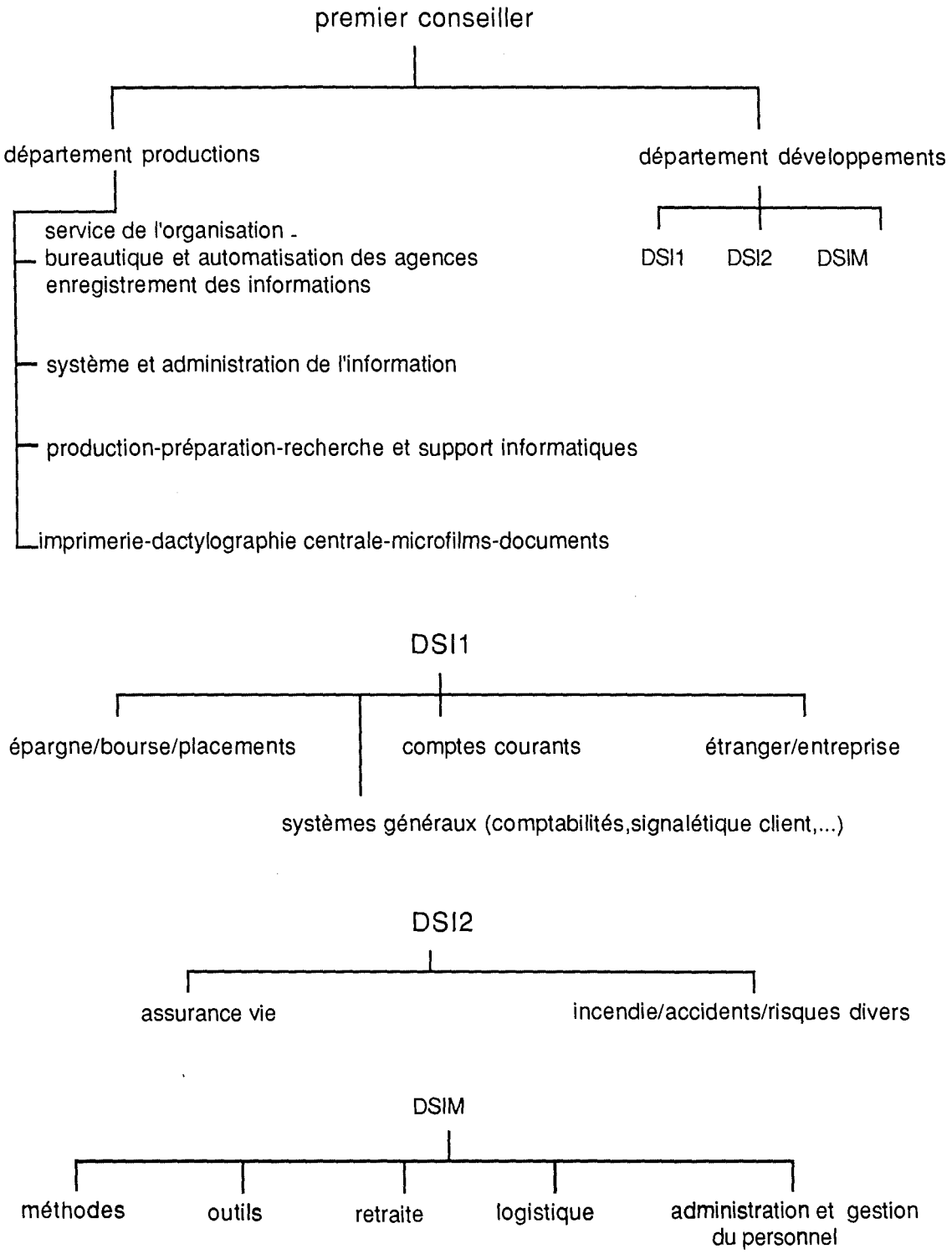


Figure 1.2: structure du département informatique de la CGER.

Le département informatique est subdivisé en deux grands pôles: les productions et les développements.
La production renvoie au fonctionnement du centre de calcul et aux modes d'utilisation des ordinateurs auxquels ont été rattachés des services adjacents tels l'imprimerie et la dactylographie.
L'horizon temporel de la production est fort limité; elle a pour préoccupation "que cela tourne chaque jour" et "que chaque jour on ait atteint les objectifs prévus".
Le département développements s'occupe de la conception et de la réalisation de l'ensemble des programmes nécessaires à tous les départements de la CGER, il s'occupe également de la recherche et de la mise en oeuvre des méthodes et outils de développement de logiciels.
Comme l'indique le schéma, le département développements est lui-même subdivisé en trois cellules de Développement de Systèmes d'Information (DSI).

le DSI1 développe des applications pour l'entité 1 (banque)
le DSI2 développe des applications pour l'entité 2 (assurances)
le DSIM développe les applications communes aux deux entités (systèmes de pension, logistique, administration des ressources humaines), il comprend également les groupes méthode et outils.

Chaque DSI est composé de divers groupes de développement orientés chacun vers un ou plusieurs produits de la CGER.
Chaque groupe de développement important est composé d'un éventail complet des grades existant dans le département informatique.
Ces trois grandes catégories de personnel (ou grades) sont :

- les informaticiens. Quatre classes d'informaticiens existent:
  - les informaticiens D (en général des chefs de département)
  - les informaticiens C (en général des chefs de service,coordinateurs de projet)
  - les informaticiens B (en général des chefs de projet ou des responsables d'un groupe de développement)
  - les informaticiens A (en général des chefs de projet).
- les analystes.
- les programmeurs.

Nous pouvons remarquer que les fonctions d'analyste et de programmeur restent séparées.
L'attribution des postes se fait en fonction du profil de chacun. Elle sera souvent la résultante d'un compromis ancienneté/ expérience.
La composition en terme de grades professionnels et le nombre d'employés dans la plupart des groupes de développement varient de façon importante

en fonction du type de projet sur lequel ils travaillent et du stade de développement atteint. Pour le groupe outils par exemple, la composition actuelle qui est de trois informaticiens, deux analystes et deux programmeurs va évoluer vers plus d'analystes et moins de programmeurs car ce groupe va aborder, en collaboration avec le groupe méthodes, l'étude de la mise en oeuvre d'un support de développement (AD/CYCLE) qui requiert un aspect étude de très haut niveau.
L'environnement des groupes de développement est fortement dynamique et imprévisible à long terme.
A la CGER, les groupes de développement sont tributaires des politiques des produits bancaires et des politiques des produits d'assurances dont la détermination leur échappe. Ils ne peuvent prévoir à moyen et parfois même à court terme les demandes de développement qui leur seront soumises .

Nous pouvons donc constater que:

- le critère de regroupement de base des unités est le produit.
  Chaque groupe de développement a un responsable qui, vu la taille limitée des groupes, pourra superviser celui-ci de manière efficace.
  De plus, comme le souligne H. MINTZBERG: "*généralement, les gens qui travaillent côte à côte dans de petits groupes s'adaptent les uns aux autres de façon informelle . L'ajustement mutuel est alors le principal mécanisme de coordination* " (1982, p.378).
- l'environnement est complexe et dynamique.
  Selon P. SIMULA (1986) : " *le problème des informaticiens est celui de l'adaptation continuelle au changement: changement de matériel, changement des relations entre informaticiens et vis-à-vis des utilisateurs, changement d'outils et de méthodes informatiques, changement des contraintes de productivité* " (p.159).
  Comme le note H. MINTZBERG: " *plus l'environnement est dynamique, plus la structure est organique* " (1982, p.248).
  Une structure rigide suppose un degré de stabilité et de calme.
  D'autre part, la stabilité va à l'encontre de l'adaptation.
  Les groupes de développement sont de petites unités peu formalisées (peu rigides) qui sont capables de s'adapter rapidement aux demandes et sont ainsi armées pour faire face à un environnement instable et changeant sans porter préjudice au fonctionnement harmonieux du département informatique.
  Qui suppose structure dynamique suppose également variation importante du nombre et de la composition de ses employés.

1.2.2. Les groupes d'étude

Nous avons déjà pu remarquer que le département est divisé en trois DSI eux mêmes divisés en groupes de développement orientés produits.
A côté de ces structures permanentes existent des structures temporaires: un groupe sera d'abord formé par projet pour effectuer l'étude de l'existant, pour déterminer les fonctionnalités du nouveau système et pour effectuer une première division en sous-systèmes. D'autres groupes (que nous appelerons groupes de projet) sont créés lorsqu'un développement de logiciel démarre et sont dissolus lorsque le groupe a atteint l'objectif qui lui a été assigné. Il est composé (selon le stade de développement) d'un chef de projet, de représentants du client (départements demandeurs), des Data Base Administrators, d'informaticiens, d'analystes et de programmeurs d'un ou de plusieurs groupes de développement (lorsqu'il s'agit d'une application touchant à plusieurs produits de la CGER).
Le département opte pour une structure matricielle à la fois orientée projet et orientée spécialités. Des entités se créent pour mener à bien un projet (sous la responsabilité d'un responsable) tout en faisant appel à des personnes issues de divers groupes ou services.

P. SIMULA (1986) à propos des groupes d'étude déclare: "*les découpages d'activités ne reposent plus sur l'existence de frontières strictes et rigides, ils semblent au contraire impliquer une certaine fusion entre les travaux de conception, d'analyse et de programmation, fusion facilitant par ailleurs la mise en oeuvre des méthodes modernes de programmation (programmation structurée par exemple)* " (p.107).
H. MINTZBERG (1982) souligne: "*la structure matricielle paraît être un mécanisme très efficace pour développer des activités nouvelles et pour coordonner des interdépendances multiples et complexes, cette structure n'est pas destinée à ceux qui ont besoin de sécurité et de stabilité* " (p.167).
Le développement d'un projet est en effet une "activité nouvelle" et les interdépendances qui vont se créer entre futurs utilisateurs, informaticiens, DBA, analystes et programmeurs au cours des différents stades de développement sont complexes et multiples.
Dans sa description de l'adhocratie, H. MINTZBERG déclare qu'elle tend à utiliser à la fois le regroupement par fonction et le regroupement par projet dans une structure matricielle. "*Les différents spécialistes doivent joindre leurs forces dans des équipes multidisciplinaires créées chacune pour un projet.* ".
Les groupes de projet de la CGER sont des structures matricielles temporaires auxquelles on assigne un objectif particulier et qui sont composés de divers spécialistes de différents domaines.

Enfin, SAYLES (1976) montre que la structure matricielle est faite pour les organisations qui sont préparées à résoudre leurs conflits par négociation informelle entre égaux plutôt qu'en recourant à l'autorité formelle (p.4). Le point traitant de la résolution des conflits (voir infra point 1.3.6) va dans cette direction.

1.2.3. Les relations entre les trois DSI

Chaque semaine, une réunion des coordinateurs de projet, des chefs de département et du premier conseiller débat entre autre des multiples problèmes pouvant survenir à l'intérieur ou entre les DSI (problèmes d'utilisation de base de données, d'utilisation d'un outil de développement,...). Tous les participants y réfléchissent à des questions qui ne les concerne pas toujours directement, mais qui pourraient leur servir d'expérience si un problème similaire se présentait chez eux dans le futur. A côté de ces réunions et, en fonction des besoins, des démarches plus ponctuelles et moins formelles existent. On peut ainsi arriver à des réunions pluridisciplinaires fréquentes regroupant des spécialistes de différents domaines pour fixer les protocoles, les solutions, les actions à accomplir, ...

1.2.4. Relations utilisateurs/informatique

Les steering committees servent d'interface entre les groupes de développement et leurs clients.
Il s'agit de réunions ayant lieu tous les mois et regroupant les représentants de haut niveau des utilisateurs (directeurs, premiers conseillers, sous-directeurs, chefs de service) et des représentants de l'informatique (à partir des chefs de projet). Ces comités essaient de définir ou de clarifier les besoins des utilisateurs, de considérer les états d'avancement des projets en cours, d'établir les ressources à octroyer.
A côté de cette structure, de nombreux contacts informels ont lieu. L'ampleur de ceux-ci dépend des projets à développer. Nous examinerons plus en détail les raisons et les conséquences de ces contacts informels lors du deuxième chapitre.

1.3. Moyens de coordination

1.3.1. La stratégie informatique de la CGER

Le département informatique de la CGER tient un rôle particulier puisqu'il est au service de tous les autres départements. Sa stratégie est composée de "politiques informatiques" à long terme (méthodes de développement, outils, ...) et des stratégies de ses différents clients. Les stratégies des différents départements sont élaborés au sein de structures appelées comités d'ordonnancement. Ces comités réunissent le directeur informatique ainsi que les directeurs des autres départements. Cette structure établit entre autre les divers projets informatiques sur base des dates de disponibilité souhaitée des produits, en fonction d'un compromis ressources/priorités/perspectives. Les stratégies des divers groupes de développement seront donc celles des produits pour lesquels ils développent des logiciels. Toutefois, la forte croissance des marchés financiers, l'apparition de produits financiers de plus en plus complexes et une concurrence nationale et internationale accrue obligent les institutions financières à réagir fort rapidement lorsqu'une opportunité apparaît soit en créant soit en adaptant un produit financier. De plus en plus, les produits financiers ne pourraient avoir de stratégie car ils sont des réponses à la concurrence ou des réponses à une opportunité qui est apparue. Les comités d'ordonnancement peuvent tout au plus établir les lignes directrices jusqu'à un horizon de cinq ans, il s'avère pratiquement impossible de pronostiquer l'évolution du marché au-delà. Avec l'ouverture prochaine des frontières, nul ne peut prédire par exemple comment se présentera le produit assurance-vie demain.
Le département informatique dispose aussi de ce qu'on pourrait appeler une stratégie propre, qui elle s'étend sur un horizon beaucoup plus étendu. Exemples: méthodes de développement de logiciels, dictionnaire de données, générateurs automatiques de programmes, politique d'installation du parc de PC etc, . . .)
Les options informatiques sont prises pour une période de dix ans au moins. Cet état de fait est principalement dû à deux raisons:

- aux lourds investissements engendrés par ces options. Ex: la composition d'un parc de 1000 PC suite à une décision de décentralisation du travail coûte fort cher
- les mentalités du personnel ne sont pas toujours prêtes à accepter rapidement le changement. Ex: il a fallu plusieurs années pour que tout le monde reconnaisse l'utilité de la phase d'analyse conceptuelle, et, il a fallu plus longtemps encore avant qu'elle n'ait été appliquée systématiquement.

Le département possède une large marge de manoeuvre en ce qui concerne la détermination de sa stratégie propre car ses possibilités de négociation avec la direction sont étendues.
Si le rôle du département informatique semble minime dans l'élaboration des stratégies produits, il ne faut pas en conclure que toute influence sur ces produits lui échappe et cela pour deux raisons principales:

- "*le renforcement du rôle du service informatique en matière d'organisation de bases de données, d'architectures de systèmes et de choix de matériel contribue à la redistribution du pouvoir dans l'entreprise. De technicienne, la position de l'informatique devient stratégique* " (P. SIMULA, 1986, p.119). L'informatique restant un domaine très spécialisé, le département informatique de la CGER, comme la majorité des départements informatiques, possède un grand pouvoir de proposition.
  En outre, pour chaque développement, l'informaticien est appelé à ouvrir l'éventail de ses activités à la fois dans le domaine technique et dans le domaine d'application où son rôle peut s'étendre de l'analyse des besoins aux propositions d'organisation et de réorganisation.
- des responsables de produit peuvent, suite à une relation de confiance et de reconnaissance de compétence acquise lors des développements de logiciels antérieurs, venir consulter informellement des informaticiens à propos de questions concernant la gestion de leur produit.

Nous pouvons ajouter pour conclure que, comme le département développements ne peut pas prévoir les demandes utilisateurs futures, la stratégie informatique ne se stabilise jamais, elle change sans cesse en fonction des projets qui lui sont attribués. Si la stratégie se stabilisait, le département perdrait sans doute sa capacité d'adaptation rapide et mettrait ainsi en péril la rentabilité de la CGER tout entière.

1.3.2. Les plans informatiques

Les plans informatiques sont élaborés annuellement par les comités d'ordonnancement. Jusqu'il y a peu, ils prenaient uniquement en compte le plan d'achat du matériel. A l'heure actuelle, ils comprennent également les plans de développement de logiciels qui doivent être accomplis au cours des deux ou trois années à venir. Les plans reflètent plus une réponse aux besoins des divers départements s'occupant des produits de banque ou d'assurances que de prise en compte de besoins propres au département informatique.

1.3.3. La coordination entre groupes de développement

Les comités d'ordonnancement, en assignant les développements aux différents groupes de développement, établissent dans les grandes lignes leur coordination puisque chacun d'eux est fixé sur les tâches qu'il aura a accomplir. Il se peut toutefois que, pour de multiples raisons, un groupe de développement ait besoin de l'aide d'autres groupes. Dans le cas du développement d'une application pluridisciplinaire par exemple, le chef de projet responsable identifiera les interfaces nécessaires avec les autres groupes et il prendra alors personnellement contact avec les responsables de ces autres groupes. Un ou plusieurs spécialistes pourront ainsi rejoindre le groupe d'étude.
Si les besoins de coordination sont plus temporaires, la coordination se fera par des réunions plus ou moins informelles qui, si le besoin s'en fait ressentir, peuvent se formaliser.
A l'intérieur d'un groupe de projet ou d'un groupe de développement, l'ajustement mutuel semble être le mécanisme de coordination principal. La standardisation tout comme la supervision directe, sont inadéquates vu la complexité et l'interdépendance très forte des travaux de chacun.

Comme le note KHANDWALLA (1976) : " *le travail de coordination n'est pas laissé à un petit groupe de responsables, mais assumé par la plupart des membres de l'organisation, de façon assez semblable à ce qui se passe dans une équipe de hockey bien intégrée où les membres associent spontanément leurs efforts de manière à garder les activités de l'équipe centrées sur leur objectif qui est la victoire* " (p.10).
lors d'un développement de logiciel, tous les membres du groupe de projet vont fortement interagir et vont mettre en commun leurs diverses spécialités pour arriver ensemble au meilleur résultat possible compte tenu des contraintes qui leur sont imposées.
MINTZBERG souligne que les cadres dirigeants doivent être des maîtres en relations humaines capables d'utiliser la persuasion, la négociation, la coalition et tout ce qui peut amener à rassembler les individus en équipes multidisciplinaires fonctionnant sans à-coups (1982, p.388).
Au sein des divers groupes d'étude, le chef de projet devra jouer le rôle de catalyseur pour garantir que le projet soit terminé à temps, dans les limites budgétaires fixées et dans le respect des spécifications.

1.3.4. Autonomie des groupes de développement

Un groupe de développement n'a pas d'autonomie en ce qui concerne le choix des projets: il est au service de ses clients. Ses clients, par les instances des comités d'ordonnancement, ont manifesté un besoin; le comité va alors estimer ce besoin et évaluer la priorité du projet par rapport à d'autres projets. S'il donne son accord pour le démarrage du projet, il lui attribuera un budget et un coordinateur de projet (niveau chef de service en général). A partir de ce moment, et durant toute la durée du développement, le coordinateur de projet disposera d'une large autonomie: il va décider à quel moment il aura besoin de quelles ressources, il va assurer la découpe en tâches, il va désigner un chef de projet, ... bref il s'occupera de toute la gestion du projet avec l'utilisateur, mais avec une autonomie importante vis-à-vis de la direction informatique.
Chaque année, un plan informatique doit être établi ou ajusté par chaque groupe de développement; les objectifs du groupe doivent être négociés avec la hiérarchie. Tous les six mois, des rapports d'activité (états de la situation) sont établis.
L'autonomie des groupes de développement en ce qui concerne l'achat de matériel et de logiciels est plutôt une autonomie de proposition: ils doivent s'intégrer dans la politique de la CGER; en fonction des besoins, le coordinateur établit les différentes possibilités d'achat et y ajoute une estimation des coûts et des ressources nécessaires, il présente ensuite ce rapport à la direction en indiquant sa préférence.
L'informatique se positionne encore sur un créneau essentiellement technique et fort spécialisé, ce qui lui donne un pouvoir de proposition énorme.
Cependant, les responsables informatiques doivent de plus en plus dans leurs choix se soucier des aspects économiques et sociaux.

1.3.5. La standardisation du travail du personnel

Deux points de passage sont actuellement obligatoires lors d'un développement de logiciel: la modélisation conceptuelle des données qui est contrôlée par l'étape PALOMA et le passage en production.
A part cela, le personnel bénéficie d'une liberté d'action assez importante (les consignes à respecter ne sont pas bloquantes), ce qui implique que, faute de règles précises, il devra recourir à l'ajustement mutuel pour accomplir sa tâche. Certains développeurs regrettent parfois cette liberté d'action, ils voudraient plus de directives quant aux tâches à effectuer.
Le nombre peu élevé de points de passage s'explique en grande partie par le fait qu'il est extrêmement difficile voire impossible de prouver l'efficacité éco-

nomique de telle ou telle étape et également par le fait qu'il est difficile de formaliser un point de passage (quels critères employer ?). Si nous prenons l'exemple du diagramme de flux de données, une aide est accordée à l'informaticien désirant l'adopter, mais l'imposer supposerait une validation justifiée: comment juger la validité d'un diagramme, comment prouver que l'utilité d'un tel diagramme vaut le temps nécessaire à son élaboration ?
Pour cette raison, un point de passage ne sera rendu obligatoire que lorsque les différentes parties concernées seront convaincues de son utilité et auront les moyens de l'implanter.
Le personnel est jugé individuellement par ses supérieurs hiérarchiques sur la qualité de son travail (et non pas uniquement sur la performance pure).
Ces contrôles de qualité restent toutefois timides faute de moyens pour être efficaces et ne sont pratiquement jamais pénalisants, une exception étant la mise en production.

Comme nous avons pu le constater, la structure du département est organique. Celle ci est caractérisée par des relations de travail souvent informelles (on discute plus que l'on applique des règles bien fixées) et les problèmes sont réglés à mesure qu'ils surgissent. Contrairement aux structures bureaucratiques basées sur la standardisation, la structure organique est fondée sur l'ajustement mutuel.
H. MINTZBERG (1982) souligne:" *parce que l'adhocratie cherche à innover, ses spécialistes doivent interagir de manière informelle dans des groupes organiquement structurés... . L'organisation innovatrice ne peut s'appuyer sur aucune forme de standardisation pour coordonner ses activités, elle doit éviter la division poussée du travail, les comportements très formalisés qui conduiraient à une rigidité excessive* " (p.380).

1.3.6. La résolution des conflits

La résolution des conflits se fait au niveau le plus bas possible pour ne pas surcharger les niveaux supérieurs de la hiérarchie de problèmes mineurs. Lorsqu'un conflit surgit à l'intérieur d'un groupe de développement, une discussion s'engage afin de réconcilier les protagonistes. Une discorde entre un analyste et un programmeur sur un point particulier d'une analyse par exemple se règlera au sein du groupe avec, si nécessaire, l'aide de l'informaticien.Lorsqu'un groupe de projet éprouve des difficultés à propos d'une consigne de l'utilisateur, il entamera une discussion avec lui et, si aucun accord n'est possible, le problème sera évoqué lors du steering comittee suivant et un dialogue s'y engagera.
Un conflit entre des groupes de développement différents sera résolu de la même manière. Ce n'est que lorsque les parties intéressées n'ont pas pu se

réconcilier au cours d'une réunion (formelle ou non) que la hiérarchie sera appelée à trancher le différend sur base d'un rapport de réunion contenant les arguments de tous les protagonistes.
La résolution des conflits est donc réglée au cas par cas, aucune règle écrite n'existe à ce sujet.
La résolution de conflits engendrés par l'introduction d'une méthode de développement de logiciels sera abordée ultérieurement.

1.4. L'adhocratie

1.4.1. Le département informatique de la CGER se rapproche-t-il d'une structure adhocratique ?

A ce point de notre description du fonctionnement et de la structure de département informatique de la CGER, il serait intéressant de vérifier notre hypothèse mentionnée plus haut: pourrait-on rapprocher le département développements de la structure adhocratique d'H. MINTZBERG.
Reprenons les caractéristiques de l'adhocratie applicables au département informatique.
Contrairement à la bureaucratie, le mécanisme de coordination principal de l'adhocratie est l'ajustement mutuel et non pas la standardisation des procédés de travail: vu l'interdépendance des travaux des futurs utilisateurs, des spécialistes de l'automation (informaticiens, analystes, programmeurs, spécialistes système), du personnel de la préparation et de la production, du personnel des groupes outils et méthodes, des Data Base Administrators et vu la complexité des tâches à accomplir, ces différentes personnes doivent s'ajuster mutuellement au sein d'un groupe de projet et parvenir ainsi à une collaboration collective.
Le département développements a développé des mécanismes pour encourager les contacts entre individus et ces mécanismes ont été incorporés à la structure formelle: il s'agit principalement des groupes d'étude. La présence de groupes d'étude est une autre caractéristique de l'adhocratie.
Le département applique également la décentralisation sélective: le choix des projets est déterminé par les comités d'ordonnancement, le coordinateur de projet gère le développement et les développeurs ont la responsabilité de mener à bien toutes les étapes qui les concernent dans ce développement.
Le système technique employé dans le département est complexe: les outils, méthodes et matériels utilisés par les développeurs sont complexes.
La structure organique, le regroupement des unités sur base de marchés et l'environnement complexe et dynamique sont d'autres caractéristiques de

l'adhocratie, nous avons vu plus haut qu'elles s'appliquent au département informatique de la CGER.

Le département informatique de la CGER se rapproche dons très fort de la structure adhocratique définie par H. MINTZBERG même si un ordre hiérarchique est établi et si les objectifs à atteindre sont fixés.
Il s'agit selon lui de la seule structure permettant l'innovation perpétuelle.
Cette structure n'est cependant pas parfaite. Nous allons essayer de reprendre ses principaux inconvénients.

1.4.2. Le problème de l'efficience

H. MINTZBERG (1982) souligne qu'aucune structure n'est mieux adaptée à la résolution de problèmes complexes et peu structurés. Aucune ne peut rivaliser avec elle pour l'innovation sophistiquée. Ni malheureusement par le coût de cette innovation (p.400).
" *L' adhocratie n'est simplement pas une structure efficiente* " (p.400).
Alors qu'elle est idéalement adaptée pour le projet unique, l'adhocratie n'est pas compétente pour faire des choses ordinaires (p.401).
KNIGHT (1976) explique la raison de cette inefficience: " *la racine de l'inefficience de l'adhocratie est le coût élevé des communications* " (p 126).
Lorsqu'il y a une décision à prendre dans une bureaucratie, quelqu'un donne un ordre auquel ses subordonnés doivent se soumettre; dans une adhocratie ce n'est pas le cas: tout le monde est impliqué: responsables de projet, clients, spécialistes de tous horizons et de tous niveaux hiérarchiques. Cela implique un nombre communications très élevé.
Les interactions entre les divers membres d'un groupe de projet sont en effet fort importantes: les informaticiens établissent l'étude de l'existant grâce au concours de l'utilisateur, les analystes commencent leur travail sur base des études établies par les informaticiens, les programmeurs traduisent en un langage de programmation les algorithmes établis par les analystes, les DBA doivent être consultés pour les aspects concernant les bases de données, les gens de la sécurité et de la production viennent apporter leurs connaissances,... . Toutes ces interrelations ne peuvent être supervisées par un seul membre. Tous les participants au développement d'un projet communiquent souvent de manière informelle, non canalisée, d'où un nombre de communications très élevé.

1.4.3. Le danger du changement de structure inapproprié

Une solution aux problèmes de l'inefficience réside en un changement de structure. L'adhocratie pourrait être dans ce cas transformée en une structure

plus stable et plus bureaucratique où les moyens de coordination principaux passeraient de l'ajustement mutuel à une plus grande standardisation des communications, où le travail de chacun serait défini plus précisément au moyen de règles écrites.
H. MINTZBERG (1982) souligne les dangers que comporte ce clivage:" *l'adhocratie administrative existe pour innover dans sa propre industrie. Les conditions de dynamisme et de complexité qui exigent l'innovation sophistiquée sont généralement communes à tout le secteur d'activité. L'adhocratie ne peut choisir ses clients, elle est au service de toute l'organisation. Une conversion vers une forme plus bureaucratique peut détruire sa capacité d'innovation et ainsi mettre en péril toute l'organisation"* (p.402).
Comme le relatait un informaticien: " *les informaticiens ont besoin de souplesse, ils n'aiment pas être trop guidés".*
Soulignons également que le département informatique a derrière lui tout un passé de non autoritarisme. Le passage à une forme de management plus stricte, laissant moins d'autonomie aux membres, serait très mal ressenti par la quasi totalité d'entre-eux.

1.5. Une méthode de développement de logiciels

1.5.1 Vers une formalisation plus grande du comportement

Les organisations formalisent le comportement pour en réduire la variabilité, et en fin de compte pour le prédire et le contrôler. L'organisation bureaucratique complètement formalisée est une organisation où, à chaque moment, chacun sait ce qu'il doit faire et comment il doit le faire. Cette formalisation permet de clarifier le rôle de chacun et, si les règles ont été choisies judicieusement, elle engendre une rentabilité importante.

1.5.2. Vers une rationalisation des activités

P. SIMULA (1986) note que l'évolution des emplois dans le secteur informatique est liée à la généralisation des méthodes et des outils destinés à faire passer l'informatique du stade artisanal au stade industriel.
Ce désir d'arriver à une "informatique industrialisée" semble prioritairement guidée par des contraintes économiques de rentabilité du matériel et de productivité des hommes (p.103).
Le génie logiciel regroupe toutes les méthodes et outils s'y rapportant.
Il concerne toutes les phases de développement d'un logiciel de la demande de développement à la maintenance.

P SIMULA rajoute: " *la cellule système pénètre ainsi au sein des autre fonctions: c'est elle qui informe, conseille et guide les analystes-programmeurs pour la mise en oeuvre de logiciels d'aide et éventuellement pour l'utilisation des langages et des méthodes de programmation*" (p.103).
A la CGER, les cellules outils et méthode s'occupent du génie logiciel. Elles conseillent les développeurs qui ont des questions ou des problèmes concernant un outil, une technique ou un point précis de la méthode de développement. Elles organisent aussi des cours de formation aux outils et aux méthodes et techniques de développement. Le groupe méthodes enfin valide les modèles conceptuels de données et les résultats de PALOMA.

1.5.3. Vers une plus grande productivité

L'efficacité économique qui semblait jusqu'il y a peu épargner les départements informatiques des entreprises s'est traduite par une recherche des gains de productivité en "*améliorant la compétence technique des équipes tout en réduisant les coûts de réalisation et de maintenance*" (P. SIMULA, 1986, p.121). Durant les années 60 et 70, les entreprises réalisaient des marges bénéficiaires importantes ce qui n'incitait pas à la rationalisation du travail. La crise économique eu pour conséquence de diminuer les marges bénéficiaires et d'accroître la concurrence. La hiérarchie des entreprises prit peu à peu conscience que les charges informatiques commençaient à peser fort lourd dans son budget. " *L'accroissement de la productivité du travail s'imposait donc comme une nécessite économique et financière tout en apparaissant comme une solution à la carence d'informaticiens sur le marché du travail.* " (SIMULA, 1986, p.122).
P. SIMULA souligne également que les effets de la rationalisation portaient aussi bien sur la production de logiciels que sur la mise en oeuvre de structures organisationnelles plus efficaces, conduisant par exemple à créer des filiales ou des cellules d'assistance technique et à ménager des relais entre techniciens et utilisateurs de l'informatique (p.122).
Néanmoins, il tempère son optimisme: " *en pratique, la généralisation des ces méthodes se heurte à une forte inertie. Le poids des acquis et habitudes de travail, la lourdeur de la maintenance des logiciels anciens constituent autant de freins à la rationalisation et à la diffusion des innovations du génie logiciel*" (p.123).
L'introduction d'une méthode telle que SDM dans l'environnement de la CGER doit tenir compte des facteurs existants: de la liberté des développeurs quant au choix d'une manière de travailler, des logiciels de support, des comportements peu formalisés,... . Une méthode de développement ne peut faire abstraction de tout le passé même si, du point de vue économique, ce serait plus efficace.

1.5.4. SDM à la CGER

La CGER ayant un passé de stratégie managériale résolument orienté "autonomie responsable", il n'était pas question d'imposer du jour au lendemain une méthode de développement de logiciels très précise et donc de changer complètement les habitudes de travail.
Les employés y sont considérés comme des professionnels qui connaissent bien leur métier. S'ils ont une approche de développement, cela doit être dû à une expérience assez longue ou à d'autres raisons valables. De plus, une approche très directive doit avoir un support permanent de toute la hiérarchie et les gens qui s'occupent de la méthodologie risquent d'être transformés en "policiers répresseurs".
Cette approche est considérée comme négative: le département informatique préfère motiver ses employés et faire confiance en leurs qualités de professionnels pour les amener ainsi à une solution réfléchie plutôt que d'introduire une méthode de développement contraignante qui risque fort de brimer leur créativité. La souplesse de l'informatique permet d'accorder une marge de manoeuvre assez large qui ne serait pas envisageable dans une usine par exemple. Un aspect négatif de cette souplesse réside dans le fait qu'un travail "artisanal" est économiquement moins efficace. L'aspect positif de cette souplesse est une préservation de l'autonomie, de la créativité, et des responsabilités des employés; ce qui crée une ambiance de travail plus agréable où les conflits sont rares et où chacun est reconnu comme professionnel.
L'implantation de SDM à la CGER a essayé d'établir le meilleur équilibre entre les deux aspects décrits ci-dessus.
Les méthodologues ont élaboré à cet effet un "tableau de bord" qui, pour chaque activité d'une phase de développement SDM, décrit quelle personne il faut contacter, quels outils utiliser et quels documents rédiger.
SDM a été choisie car elle est souple: elle est employée comme "checklist " sur laquelle viennent se greffer des outils connus et employés dans le département. Dans le futur, il est probable que sur base de la "checklist" SDM, la formalisation et les points de passage obligatoires seront étendus du seul modèle conceptuel normalisé à l'ensemble du cycle de vie de la construction d'une application.
Au début, deux types de réactions extrêmes furent constatées:

- les employés qui avaient auparavant émis le désir de disposer d'une méthode afin de faciliter leurs tâches ont accueilli SDM de manière favorable en espérant qu'elle allait constituer une espèce de toboggan où il suffirait de se laisser glisser et, qu'une fois arrivé à son pied, l'application serait terminée.
  SDM ne va pas rencontrer les desiderata de ces personnes puisqu'elle

laisse une marge de manoeuvre importante, elle ne va pas guider totalement les développeurs en précisant quoi faire à chaque instant. Une méthode de développement qui satisferait pleinement ces développeurs n'existe d'ailleurs pas (encore).

- les employés qui ne désiraient pas de méthode avant l'introduction de SDM en eurent au départ une appréciation critique: ils ont considéré SDM comme frein à leur autonomie, comme une lourdeur administrative bref comme une perte de temps.

Ces conflits peuvent être considérés comme une maladie de jeunesse que connaît toute méthode au sein d'une organisation. Ils s'atténuèrent d'ailleurs au cours du temps car SDM n'avait pas pour objectif de freiner la créativité des employés, mais, au contraire, de leur faciliter le travail en leur suggérant de bien se poser toutes les questions et de bien passer en revue toutes les étapes de développement.

Le pas le plus difficile qui restait alors à franchir était la prise de conscience par les chefs de projet que la méthode n'était pas destinée à les brimer, mais qu'elle constituait le moyen le moins mauvais possible d'arriver à des résultats de qualité. Les initiateurs de la méthode restèrent d'ailleurs toujours à l'écoute des propositions émises par les chefs de projet.

Comme les méthodes de développement de logiciels en sont encore à leur début et que leur application dans les entreprises est un phénomène relativement neuf, nous ne pouvons pas affirmer que SDM fait partie des meilleures, mais, pour ce qui est de la CGER, elle répond au besoin de standardisation et elle est surtout de plus en plus reconnue, acceptée et appliquée par les développeurs. Cette reconnaissance a été très lente car elle est la résultante d'une concertation avec à peu près tous les développeurs et non pas la résultante d'une obligation d'emploi.

Même si SDM est une méthode souple, elle constitue inévitablement un frein à la créativité, mais ce frein s'applique uniquement à la créativité de l'exécution et non pas à la créativité de conception. Elle aide par exemple les développeurs à ne pas oublier une étape qui risque d'être pénalisante pour eux par la suite (procédures de sécurité, réorganisation éventuelle du processus de travail,...).

Le groupe méthodes n'a pas d'autorité formelle sur les développeurs. Les développeurs appliqueront SDM uniquement lorsqu'ils seront convaincus de son utilité. Une obligation d'application de la méthode sera peut être instaurée un jour, comme cela s'est produit pour l'analyse conceptuelle, lorsque SDM sera, pour les développeurs, devenue synonyme d'aide efficace pour leur travail et sera à leurs yeux devenue transparente (grâce notamment aux outils de développement).

En résumé, l'influence qu'exerce SDM sur la créativité des développeurs tient en deux points:

- la créativité de réalisation s'est standardisée
- la créativité de conception a été renforcée

Les développeurs restent très créatifs dans la solution qu'ils vont mettre en oeuvre, mais ils sont guidés dans la manière dont ils vont développer ces solutions. Cela aura pour conséquence une amélioration de rentabilité.
Après quelques mois d'application de SDM, les résultats quant à la qualité des logiciels développés selon ses normes se sont avérés positifs.
Il est trop tôt encore pour établir l'impact de la méthode sur l'efficience des développeurs et sur la maintenance des logiciels.

CHAPITRE 2
RESULTATS DES INTERVIEWS

Ce chapitre se base sur les interviews menées au sein du département informatique.

2.1. Présentation de l'enquête.

Décrivons brièvement la méthode d'interview qui a été utilisée.
Les interviews ont été menées durant 9 jours dans le département informatique de la CGER à Bruxelles. Chaque interview avait une durée moyenne d'une heure.
Durant ces 9 jours, nous avons interviewé 32 membres du département. et un utilisateur.
L'échantillon des interviewés a été déterminé par le chef de service du DSIM et par un méthodologue. Il a été choisi de telle manière à contenir des personnes de tous grades professionnels, appartenant à divers groupes de développement et étant représentatives des différents degrés d'application de SDM.
Les développeurs interviewés appartiennent à divers groupes de développement. En voici la liste:

- assurance-vie
- assurances diverses
- bourse
- comptes universels
- étranger
- épargne
- pension
- logistique
- systèmes généraux
- personnel.

De plus, des membres des services de support suivants ont été interviewés: sécurité, DBA, méthode, outils, préparation, production.
Les items analysés peuvent se classifier en trois types: la situation personnelle des interviewés, l'organisation et la répartition du travail au sein de leur groupe d'étude et leur perception et application de la méthode SDM.
Les questions sur:

- la situation personnelle s'intéressent principalement :
    - au grade professionnel, à la formation, à l'ancienneté des interviewés
    - aux étapes de développement dans lesquelles ils interviennent

- aux outils, techniques employés pour l'élaboration de ces étapes et aux points de contrôle et problèmes rencontrés lors de l'accomplissement de ces étapes

- l'organisation et la répartition du travail s'intéressent principalement:
  - à la répartition et à la coordination du travail au sein des groupes d'études dans lesquels interviennent les interviewés
  - à la circulation de l'information en général
  - à l'autonomie des personnes

- la méthode SDM s'intéressent principalement:
  - à la perception des interviewés à propos de la méthode
  - aux raisons de l'application ou de la non application de la méthode
  - à l'aide que peut apporter SDM.

Le questionnaire qui a été employé se trouve en annexe 1.
Il a été envoyé à l'avance à toutes les personnes qui allaient être interviewées de façon, à ce qu'elles aient une première idée du contenu des questions et à ce qu'elles préparent, si nécessaire, certaines d'entre elles.
Lors des interviews, le questionnaire n'a pas été suivi question par question. Vu la durée limitée des interviews, il nous a semblé plus intéressant de laisser chacun répondre aux questions qu'il estimait les plus intéressantes en fonction de ses intérêts, de son expérience et de ses idées (pour les propositions d'amélioration par exemple).
Ceci explique le taux de réponse très faible à certaines questions.

2.2. Questions générales.

2.2.1. Grades professionnels des interviewés.

GRADES PROFESSIONNELS	NOMBRE D'INTERVIEWS
PREMIER CONSEILLER	1
INFORMATICIENS D (sous-directeurs)	2
INFORMATICIENS C	4
INFORMATICIENS B	4
INFORMATICIENS A	8
ANALYSTES	10
PROGRAMMEURS	3
UTILISATEUR	1
TOTAL	33

2.2.2. Quel est le type de tâches principal des interviewés?

A travers cette question, nous cherchions à découvrir en quoi consistait les activités des interviewés afin de mettre celles-ci en relation avec d'autres réponses (degré d'application de la méthode SDM, perception quant à la méthode,...).

RESULTATS:

type de tâches	
encadrement (à partir des chefs de service)	8
développement de nouvelles applications	10
maintenance d'anciennes applications	3
développement et maintenance	3
support	8
utilisateur des services informatiques	1
TOTAL	33

2.3. Questions relatives à la méthode SDM

2.3.1. Degré d'application de la méthode SDM

Cette question était tout d'abord destinée à mesurer et à classifier le degré d'application de SDM pour, par la suite, pouvoir mettre les résultats en relation avec d'autres réponses: qui applique SDM, qui n'applique pas SDM, quelles sont les raisons pour lesquelles SDM est appliquée, quelles sont les raisons pour lesquelles elle n'est pas appliquée,...

RESULTATS:

	nombre de réponses
application complète	4
application de la philosophie	5
pas d'application	3
ignorance de SDM	5
non développeurs	16
TOTAL	33

Nous avons classé les réponses en cinq catégories:

- celle qui reprend les développeurs suivant complètement la méthode càd ceux qui lors des développements passent par toutes les étapes de SDM applicables à leur projet et qui les détaillent au maximum en fonction du temps qui leur est accordé
- celle qui reprend les développeurs qui suivent la philosophie de la méthode.
  Ces personnes ne passent pas forcément par toutes les étapes de SDM applicables à leur projet, mais leur méthode de développement se rapproche fort de la méthode SDM
- celle qui reprend les développeurs ne suivant pas la méthode.
  Ces développeurs, pour des raisons diverses que nous analyserons plus tard, n'estiment pas que SDM peut leur apporter une aide, ils continuent donc à développer des applications sans tenir compte de SDM
- celle qui reprend les développeurs ne sachant pas ce qu'est SDM
- celle qui reprend les personnes dont le travail n'est pas directement concerné par la méthode. Il s'agit des chefs de service ainsi que de leurs supérieurs hiérarchiques et des personnes travaillant dans les groupes de support et n'ayant pas à développer de grosses applications. Nous y avons également repris l'utilisateur car, même si celui-ci est un interlocu-

teur précieux pour les informaticiens, il ne développe pas lui-même les applications.

Mettons ces réponses en relation avec la perception qu'ont les développeurs à propos de la méthode pour mieux comprendre ce qui les incite ou ce qui les dissuade à utiliser SDM.

degré d'application	perceptions positives		perceptions négatives	
complète	SDM est une méthode souple	2	SDM ne fournit pas assez de guidance à l'intérieur de chaque étape	2
	SDM fournit une guidance efficace lors d'un développement	2	SDM engendre trop de travail administratif	1
	la liste des étapes SDM permet de prendre en compte tous les éléments importants d'un développement	1		
philosophie	la liste des étapes SDM permet de prendre en compte tous les éléments importants d'un développement	2	SDM engendre trop de travail administratif	4
	SDM est une méthode souple	1	les outils de support ne sont pas assez intégrés	2
	SDM fournit une guidance efficace	1		
	SDM demande l'élaboration d'une bonne documentation	1	SDM n'est pas assez adaptée aux différents types de développement	1
	SDM constitue un bon coordinateur de passage des informations	1		
	SDM engendre une meilleure standardisation des logiciels	1		
pas			SDM engendre trop de travail administratif	3
			SDM n'est pas assez adaptée aux différents types de développement	2
			SDM engendre un changement des habitudes de travail	1
TOTAL		12		16

Nous interpréterons ces résultats dans la suite de ce chapitre.

2.3.2. Quels sont les principaux points forts de l'implantation de SDM à la CGER?

L'objectif de cette question était de déterminer puis d'analyser les perceptions positives des interviewés à l'égard de la méthode afin d'en déduire les raisons pour lesquelles ces perceptions ont été citées.

RESULTATS:

	nombre de réponses
la checklist laisse une grande souplesse d'utilisation	5
SDM engendre des développements plus standardisés	4
SDM fournit une guidance efficace dans le travail	4
SDM permet de prendre en compte tous les éléments importants	3
SDM laisse une grande part de créativité à l'intérieur des étapes	2
SDM constitue un bon coordinateur de passage d'information	2
SDM demande l'élaboration d'un bonne documentation	1
SDM permet de définir des points de contrôle après chaque étape	1
TOTAL	22

Interprétation des résultats.

a. La checklist permet une grande souplesse d'utilisation

Comme nous l'avons souligné dans le premier chapitre, le travail informatique requiert une large part de créativité et d'adaptation rapide. La souplesse de la méthode SDM est citée comme avantage principal. Quatre réponses proviennent d'informaticiens A, dont deux appliquent de près la méthode, un applique sa philosophie et un travaille dans un groupe de support, la dernière réponse provient d'un informaticien de la ligne hiérarchique n'étant pas directement concerné par l'application de la méthode.
La checklist permet de prendre en compte certaines étapes et pas d'autres si le projet ne s'y prête pas: certaines étapes ont déjà été effectuées, le type de projet ne nécessite pas certaines étapes, certaines étapes ont été sous-traitées,... .
La checklist permet également d'amoindrir le sentiment négatif que les développeurs peuvent avoir au départ à l'égard d'une méthode de développement qui, tout compte fait, vise à standardiser leur travail. Elle ne forme pas un bloc monolithique à appliquer dans son entièreté, elle propose au contraire une liste de points dont beaucoup sont déjà connus et réalisés par les dévelop-

peurs et qui peuvent selon les cas être appliqués ou non.
Le côté liste exhaustive permet à la méthode de s'adresser à tous les développeurs quelles que soient leurs tâches et leur spécialité. Ceux-ci pourront retrouver dans la liste les étapes qui les concernent directement.

b. La méthode permet des développements plus standardisés.

Le "vade-mecum" SDM reprend pour chaque étape de la méthode les personnes à contacter ainsi que leur numéro de téléphone, les techniques et les outils disponibles pour réaliser ces étapes. Ce vade-mecum a été très bien accueilli car la réalisation de toute application d'une certaine taille nécessite le concours d'un nombre important de personnes (utilisateurs, informaticiens, analystes, programmeurs, DBA, gens de la sécurité, gens de la production, méthodologues, ...), il constitue un cadre pour le développement des applications qui pour chaque phase indique le responsable ou la personne à contacter. SDM introduit ainsi une sorte de standardisation, de développement idéal, une guidance qui était devenue indispensable vu la multitude des intervenants et des interrelations entre les travaux de chacun d'eux.

Trois réponses du tableau proviennent de personnes n'étant pas directement concernées par la méthode (deux de la ligne hiérarchique et une d'un groupe de support) et la quatrième provient d'un analyste appliquant la philosophie de SDM.
Ce point fort de la méthode a donc principalement été relevé par des personnes dont la tâche principale n'est pas le développement ou la maintenance de logiciels, mais qui ont un intérêt dans la standardisation de ceux-ci:

- la ligne hiérarchique (à partir des chefs de service) grâce aux développements plus standardisés peut:
    * mieux planifier et contrôler les développements
    * réduire la dépendance vis-à-vis des développeurs
    * si elle le désire introduire plus de contrôle
    * plus facilement automatiser certains processus de développement .
- les groupes de support sont assurés que les développements en prenant en compte tous les éléments importants tiendront également compte de leur spécialité car la méthode préconise une participation importante de ces groupes dans les développements.

c. Guidance efficace dans le travail

Trois des quatre réponses du tableau proviennent de personnes appliquant complètement la méthode ou suivant sa philosophie.
SDM constitue donc pour eux une aide à travers la multitude de tâches qui leur incombent: SDM est une guidance, une ligne de conduite. Son application à la CGER peut être qualifiée de porte-manteaux: les phases SDM ont été soutenues par les outils qui existaient au département informatique.
Une méthode telle que SDM formalise chaque étape, elle précise quels documents doivent être établis. Une demande de développement devra par exemple être détaillée, les besoins de l'utilisateur seront analysés en profondeur:

- 1. 2 rassembler les informations sur la situation existante
- 1. 3 analyser et évaluer ces données
- 1. 4 déterminer les objectifs et exigences du nouveau système.

Après avoir défini les objectifs et les frontières de l'application avec le concours de l'utilisateur, l'informaticien chef de projet pourra être guidé par SDM afin de ne négliger aucun aspect: différentes analyses, planning global, prise en compte des problèmes de conversion et de mise en production, prise en compte des problèmes de sécurité et de confidentialité, détermination des problèmes humains, ...
Sur ces bases, les analystes et les programmeurs pourront à leur tour se servir de la méthode comme guidance pour leurs travaux.
SDM est de par sa structure "checklist" un bon coordinateur de passage d'information, chacun connaît les tâches qui lui sont imparties. SDM leur fournit la liste des documents à effectuer (documents sur base desquels travailleront les développeurs situés en aval dans le processus de développement). Certaines plaintes ont en effet été formulées quant à l'absence ou à la mauvaise qualité des analyses faites par les développeurs "amont". "*Si ces analyses n'ont pas été faites convenablement cela ne sert à rien d'appliquer une méthode".* D'autre part, certains développeurs "amont" regrettent le manque d'autorité qu'ils peuvent exercer sur leurs subordonnés: "*je ne peux pas les contraindre à utiliser une méthode quelconque*" . Ce manque d'autorité est cependant plus dû à des facteurs de personnalité qu'à des facteurs d'autorité hiérarchique.
La méthode permet de remédier ainsi aux regrets exprimés par de nombreux analystes et programmeurs concernant le manque de vue d'ensemble qu'ils ont de l'application sur laquelle ils travaillent: "*on ne sait pas à quoi sert le fruit de nos travaux*" . Si les phases amont des développements sont, comme le conseille SDM, bien documentées, ces personnes auront l'occasion, si elles

le souhaitent, d'acquérir une meilleure connaissance du projet.

d. SDM permet de prendre en compte tous les éléments importants d'un développement

Même si SDM n'a pas été employée pour définir chaque poste de travail, elle permet, grâce à la formalisation qu'elle introduit, d'éclaircir le rôle de chacun en leur proposant une description complète d'un développement. Les nouveaux développeurs se plaignent parfois du manque de précision des consignes de travail qui leur sont adressées et de la complexité de l'environnement dans lequel ils travaillent (produits, interrelations, outils, . . .). Le fait d'appliquer une méthode de développement permet de remédier à ces difficultés.
Le travail et les besoins des utilisateurs seront pris en compte de manière approfondie et formelle, les tâches et les documents que chaque développeur aura à accomplir sont formalisés, tous les aspects à prendre en compte seront examinés. Cette formalisation engendre des relations interdéveloppeurs beaucoup plus simples.

e. SDM laisse une grande part de créativité à l'intérieur des étapes

Vu l'environnement de la CGER, le groupe méthodes n'a pas voulu limiter l'autonomie des développeurs, il a élaboré un outil de travail, une énumération d'étapes à suivre, mais, à l'intérieur de chaque étape, les développeurs (principalement les informaticiens et les analystes) gardent une large marge de manoeuvre. SDM freinerait donc uniquement la liberté concernant la méthode de travail des différents participants aux développements.

f. SDM est un bon coordinateur de passage d'information

Les flux d'informations engendrés par un développement sont fort nombreux: flux entre l'utilisateur et l'informatique, entre les développeurs et les groupes de support, flux à l'intérieur des groupes de travail. Comme nous l'avons déjà souligné aux points b et c, les flux entre groupes de support et développeurs et les flux à l'intérieur d'un groupe de travail vont être quelque peu formalisés par la méthode.

Voyons ce qu'il en est de l'interface utilisateurs/informatique: les premières étapes de SDM sont:

1.1 définir le problème et le champ d'étude
1.2 rassembler les données sur la situation existante
1.3 analyser et évaluer les données
1.4 déterminer les exigences du nouveau système
1.5 arrêter les points qui restent à résoudre et les hypothèses de base.

Ces étapes se font en étroite relation avec les utilisateurs. SDM, si elle est bien suivie, permet de clarifier les besoins de l'utilisateur, d'établir une meilleure délimitation du futur système. Les utilisateurs pourront formuler des désirs plus précis (si toutefois ceux-ci ont les idées claires sur ce qu'ils veulent). L'établissement de ces frontières est fort important car beaucoup de chefs de projet se plaignent du fait que les utilisateurs ne savent pas souvent ce qu'ils veulent ou l'expriment mal: ils énumèrent un tas de cas particuliers sans pouvoir en retirer le cas général; il en découle des objectifs du futur système non exhaustifs qui vont à leur tour engendrer des demandes de modifications lorsque le projet sera déjà en phase de développement. Ces modifications vu leur aspect tardif seront extrêmement coûteuses.
Dans le meilleur des cas, certains utilisateurs suivent eux-mêmes des cours de présentation de SDM. Les effets en sont fort positifs car ils permettent d'élargir le champ de vision des utilisateurs, ceux-ci sont renseignés sur la façon dont les informaticiens travaillent, se rendent mieux compte des contraintes informatiques, des techniques utilisées, ...
La communication utilisateurs-informatique en est toujours fortement améliorée.
En ce qui concerne le passage d'informations entre développeurs, nous pouvons faire remarquer que l'information circule mieux entre le chef de projet et ses développeurs et entre les développeurs car une formalisation des études et de la documentation à établir va:

- diminuer les besoins de communication engendrés par des études ou par des analyses incomplètes
- fournir des études et des analyses (donc des informations) plus riches et plus complètes.

g. L'application de SDM engendre une meilleure documentation

La documentation est un point sur lequel insiste beaucoup SDM. Elle est d'une importance cruciale pour la rapidité et la qualité des maintenances et des ajouts ultérieurs. Des analystes se plaignent en effet du fait que, lorsque certains utilisateurs introduisent une demande de modification d'un programme assez ancien, ils sont contraints d'apporter les changements direc-

tement dans le code source faute de documentation et de spécifications.
Les développeurs travaillant avec des firmes extérieures soulignent la nécessité d'une grande standardisation de développement.
L'application de SDM permet également de résoudre le problème des mises à jour d'analyses lors des modifications: certains produits financiers progressent rapidement suite à l'évolution de la concurrence, du marché, des contraintes juridiques, . . . ce qui entraîne des modifications fréquentes des logiciels les concernant. Des programmes bien structurés et documentés au départ peuvent alors perdre rapidement leur qualité. Les schémas de départ (schéma E/R, diagrammes de flux, . . .) ne sont pas toujours mis à jour et ne correspondent plus au système réel après les modifications.
Certains informaticiens regrettent d'autre part que leurs analyses ne sont pas toujours prises en compte en aval (pour des raisons de performance par exemple); dans ce cas également, les analyses ne sont pas modifiées et il en résulte une divergence entre documentation et système réel.
Nous verrons au chapitre 3 comment SDM peut être employée pour faire face à ces problèmes.

h. SDM va permettre de définir plus facilement les points de contrôle après chaque étape

SDM en formalisant les différentes étapes d'un développement, en spécifiant les analyses et les documents à accomplir sera peut être le premier pas d'un mouvement vers un contrôle plus strict des développeurs. Cette possibilité a du moins été émise par un informaticien.

2.3.3. Quels sont les principaux points faibles de l'implantation de SDM à la CGER?

L'objectif de cette question était de déterminer puis d'analyser les perceptions négatives des interviewés à l'égard de la méthode afin d'en déduire les raisons pour lesquelles ces perceptions ont été citées.

RESULTATS:

	nombre de réponses
SDM engendre trop de travail administratif	8
SDM n'est pas assez adaptée aux différents types de développement	4
SDM ne détaille pas assez chaque étape	4
les outils de support ne sont pas assez intégrés	3
L'application de SDM engendre des modifications dans les habitudes de travail	2
TOTAL	21

Interprétation des résultats.

a. SDM engendre trop de travail administratif

La répartition des réponses est la suivante:

- 1 réponse est issue d'un développeur appliquant SDM complètement
- 4 réponses sont issues de développeurs appliquant la philosophie de SDM
- 3 réponses sont issues de développeurs n'appliquant pas SDM.

Le travail administratif engendré par SDM (études, documentation, rapports) est l'élément dissuadant le plus les développeurs à l'emploi de la méthode. SDM engendre en effet un travail administratif important.
La description du point 1. 1: *définir le problème et le champ d'étude* comporte entre autre les points suivants:

- établir les enjeux et les limites
- établir les buts de l'étude
- établir les estimations de coût et de temps
- établir la répartition du travail, les attributions et les responsabilités
- établir un plan des activités
- établir un "accord de projet" étendu. L'accord de projet reprendra:
    - la problématique (à définir à la suite d'interviews)
    - les buts du projet
    - les limites du projet dans les domaines suivants
        * temps et coûts
        * organisation
        * social
        * emploi d'outils
- les attributions
- l'établissement de rapports: à qui, selon quelle façon, à quelle fréquence

- le planning
- l'allocation du temps désiré par activité
- qui doit réaliser quoi

Les informaticiens ne disposent pas souvent du temps suffisant pour mener toutes les études et établir les documents décrits ci-dessus et applicables à leur projet. L'ampleur de la documentation que conseille SDM est importante et donc grande consommatrice de temps. Ce volume de documentation a un effet négatif sur la perception des développeurs quant à la méthode: ils auraient plutôt tendance à la considérer comme une formalité administrative freinant la rapidité de développement. Cet état de fait pourrait être résolu par des outils plus intégrés qui viendraient soutenir SDM.
A cet effet, les groupes méthode et outils vont prochainement réaliser une interface destinée à rapprocher le dictionnaire de données DATAMANAGER et l'outil EXCELERATOR de manière à permettre le transfert de données entre les deux.
Ce projet sera le premier pas vers une plus grande facilité de communication entre les différents outils de développement.

Pour revenir à la répartition des réponses, il est étrange de constater que si le travail administratif engendré par SDM dissuade les développeurs qui n'appliquent pas la méthode de près, il ne dérange pas beaucoup ceux qui l'appliquent de près. Cela s'explique peut être par le fait que les développeurs appliquant la méthode de près sont plus conscients de l'importance et de l'utilité d'une bonne documentation et sont prêts à y consacrer beaucoup de temps.

b. SDM n'est pas adaptée aux divers types de développement

La répartition des réponses du tableau est la suivante:
- 2 réponses proviennent de développeurs n'appliquant pas la méthode
- 1 réponse provient d'un développeur appliquant la philosophie de la méthode
- 1 réponse provient d'une personne travaillant dans un groupe de support.

Des trois développeurs, deux s'occupent essentiellement de maintenances et une s'occupe à la fois de maintenances et de développements .
Le caractère général et donc non adapté de SDM semble surtout gêner les personnes dont la tâche principale n'est pas le développement de nouvelles applications.
Une seule méthode est prévue pour les développements internes de toutes tailles, pour la customisation des progiciels, pour les maintenances de pro-

grammes de tous types, ... En ce qui concerne la maintenance des anciens programmes par exemple, la mise à jour des analyses est impossible faute de documentation. Une petite modification demandée par les utilisateurs ne pourra pas être effectuée en suivant rigoureusement la ligne de conduite SDM, car elle doit souvent être accomplie très rapidement, ce qui ne laisse pas le temps pour la mise à jour des études. On en arrive dans ce cas à des études et à des analyses ne correspondant plus au code, ce qui rend les utilisations futures des analyses pratiquement impossibles. D'autre part, la méthode décrit des tâches effectuées aussi bien par des informaticiens, des analystes, des programmeurs, des DBA, . . . Plusieurs d'entre eux pensent qu'une méthode adaptée sinon à chaque poste de travail devrait au moins l'être à chaque grade professionnel ou à chaque spécialité. Cela éviterait d'aller chercher dans la checklist les tâches relatives à chacun avec, comme danger, le fait que certaines étapes ne soient pas réalisées, chacun pensant qu'elles sont du ressort d'un autre.
On passerait d'une philosophie centrée sur les développements vers une philosophie centrée sur les postes de travail.
Chaque poste de travail devrait selon eux être décrit (tâches, responsables, structures de travail, outils, . . .) de manière à ce qu'ils ne doivent plus aller "pêcher" les aspects de la méthode qui les concernent.

c. SDM ne détaille pas assez chaque étape d'un développement

La répartition des réponses du tableau est la suivante:

- 2 réponses proviennent de développeurs appliquant complètement SDM
- 1 réponse provient d'un membre de la ligne hiérarchique
- 1 réponse provient d'un membre d'un groupe de support.

Des quatre développeurs suivant la méthode complètement SDM la moitié trouve qu'elle reste trop générale à l'intérieur de certaines étapes.
La caractéristique principale de l'application de la méthode dans l'environnement CGER se caractérise en effet par son aspect checklist.
Des développeurs regrettent que SDM reprenne ce qui existait déjà, elle a formalisé toutes les étapes d'un développement, mais, pour chacune d'elles, son aide reste limitée.
La volonté du département lorsqu'il a introduit SDM était de proposer une ligne de conduite à travers les développements de logiciels importants sans cloisonner le travail des développeurs et sans porter atteinte à leur créativité.
A l'intérieur de chaque étape décrite, il reste une large marge d'autonomie.
Nous prendrons comme exemple l'étape 2. 3 de l'analyse fonctionnelle: diviser le système en sous-systèmes et les décrire.

La description de cette étape est la suivante :

1) introduction.

un sous-système est une partie qui peut être développée de manière autonome.

Les raisons de la division d'un système sont les suivantes:

- complexité réduite
- meilleur contrôle
- meilleure utilisation des capacités homme

la division doit être telle que:

- les relations au sein d'un sous-système doivent être maximales (division sur base des fonctions)
- les plages (constituées de données) qui se recouvrent doivent être minimales.

outils de support

- HIPO et SADT: techniques de schéma, aide à définir les flux d'entrée et de sortie de chaque sous-système
- les diagrammes de flux de données

2) méthode de travail.

- diviser le système en sous-systèmes sur base fonctionnelle
- minimaliser les plages de recouvrement (celles-ci peuvent uniquement être constituées de données)
- décrire chaque fonction en détail

3) application à la CGER.

* technique:
    - diagrammes de flux
    - HIPO (Hierarchy Input Process Output)
    - HSAT (Hierarchical Structure of Activities and Tasks)
* sécurité:
    - lors de la division en sous-systèmes, il est nécessaire de porter attention à l'aspect "restart"
* dictionnaire de données:
    - confirmation dans le DATAMANAGER sous les member-types "SY" (system), "AC" (activity) et "TK" (task).

Source: [groupe méthodes, 1989, p.27]

La méthode de travail décrite ci-dessus démontre que pour effectuer la division en sous-systèmes, une large marge de manoeuvre (et de créativité) est laissée aux informaticiens. Des outils de support sont proposés, mais ne sont pas obligatoires. Une division en sous-systèmes dépend donc pour une grande part du savoir faire et de l'expérience des informaticiens qui s'en occupent.

Certains développeurs regrettent que SDM ait repris ce qui existait déjà (par exemple les outils) sans préciser davantage comment exécuter une étape. C'est ici l'aspect checklist de la méthode qui est critiqué: liste de tâches, mais pas vraiment de guidance à l'intérieur de chacune d'elles.
Le groupe méthodes a pris conscience du caractère trop général de la description de certaines étapes. Il envisage d'y remédier en ajoutant des éléments plus précis et plus objectifs aux étapes qui posent le plus de problèmes (indicateurs, règles chiffrées,...). Des éléments quantitatifs pourraient par exemple être intégrés à l'étape 2.3 pour permettre des découpes en sous-systèmes sur base d'éléments chiffrés (coefficients d'isolement, indices de satisfaction,...).

d. Manque d'intégration des outils de support

L'inconvénient suivant est l'interface imparfaite entre les divers outils qui viennent soutenir la méthode aux différents stades de développement.
Les réponses du tableau se répartissent comme suit:

- 2 proviennent de développeurs appliquant la philosophie de SDM
- 1 provient d'un membre de la ligne hiérarchique

Ce manque d'interface constitue donc un élément dissuasif à l'application de SDM.
Essayons de voir pourquoi.
Actuellement, la constitution de la documentation et les différentes analyses prennent beaucoup de temps et, par manque d'intégration des outils, les mêmes données doivent être introduites plusieurs fois. De même, certains de ces outils sont considérés comme lourds à utiliser (peu de possibilités graphiques, modifications lourdes, . . .). Les outils de support semblent jouer un rôle très important pour l'acceptation et l'adoption d'une méthode telle SDM, les outils CASE sont pour cette raison indispensables.
Les développements réalisés selon SDM prennent en compte tous les aspects importants pour la réussite de l'informatisation, ils sont bien structurés et bien documentés.
Le revers de la médaille est l'investissement en temps que requiert l'élaboration de cette documentation.
L'application à la lettre de la méthode requérerait selon certains un temps de développement équivalant au triple du temps consacré à un développement ne suivant aucune méthode particulière. Une simple modification dans une application engendre des changements dans les analyses de tous niveaux et exige un travail administratif très lourd et très lent. Cette lenteur n'est pas toujours acceptée par les utilisateurs.

Les causes principales de cette lenteur sont:

- les mêmes informations doivent être introduites plusieurs fois dans des outils informatiques différents (manque d'intégration des outils)
- la documentation "papier" est très lourde à élaborer
- manque d'outils pour la gestion des projets (planning de développement, gestion du temps des développeurs, facturation du projet, . . .)

L'apport de SDM à la qualité des logiciels et à la rapidité de création semble freiné par le manque d'intégration et la lourdeur des outils de support. Prenons deux exemples pour illustration :

- le dictionnaire de données DATAMANAGER n'a pas de capacités graphiques. Les informaticiens souhaiteraient des possibilités graphiques pour, par exemple, y élaborer directement le schéma E/R; actuellement, il est effectué sur papier avec toutes les difficultés que cela engendre lors des diverses modifications, et il est introduit dans le dictionnaire (de manière non graphique) seulement lorsqu'il est stabilisé.

- Le planning des projets (échéances, participants, répartition des tâches, ...) élaboré par chaque chef de projet est dans le meilleur des cas fait sur PC grâce à l'outil SUPERPROJECT qui ne communique pas avec les autres applications ayant trait au développement. Le plan de charge annuel d'un groupe de développement doit être complètement recalculé car il ne peut s'appuyer sur les données introduites tout au long de l'année.

La boîte à outils de support idéale devrait donc comprendre une série d'outils intégrés guidant les diverses phases de développement, des outils de gestion de projet, un agenda, ...
Un planning pourrait être établi à partir des tâches élémentaires et des ressources disponibles et procéder ainsi à un suivi du personnel et à un suivi des tâches.
Même si un tel outil est utopique, nous pouvons cependant remarquer que, pour pratiquement chaque phase de développement, des outils existent à la CGER, mais que les interfaces entre eux étaient jusqu'ici souvent inexistantes.
Il est important de rappeler que la mise en oeuvre d'une interface entre les différents outils est un des objectifs des groupes méthode et outils et que le premier pas de cette mise en oeuvre sera la création d'une passerelle entre le dictionnaire de données et l'outil EXCELERATOR.

La philosophie AD/CYCLE d'IBM rejoint dans les grandes lignes l'outil de support idéal décrit ci-dessus. Certains développeurs craignent que les efforts méthodologiques fournis aujourd'hui seront balayés par ce "repository" d'IBM (futur dictionnaire de données commun à tous les outils de développement) car la méthode adoptée par un tel outil sera contraignante et peut être différente de SDM; d'autres pensent que SDM sera concrétisée par le repository, celui-ci mettant des outils à la disposition de la méthode théorique SDM.

e. SDM engendre des modifications dans les habitudes de travail

Au fil des années, les développeurs acquièrent des habitudes de travail qu'il ne sera pas facile de modifier.
Les développeurs travaillant depuis peu à la CGER sont plus favorables à SDM. Non seulement, leur formation les a dans la plupart des cas initiés à une méthode de développement ou de programmation structurée, et ils estiment en outre SDM comme une aide précieuse pour s'y retrouver dans l'environnent complexe qui les entoure: la méthode constitue une guidance qui leur permet de ne négliger aucun aspect d'un développement.
Les développeurs plus anciens, d'une part, connaissent mieux l'environnement informatique ainsi que les produits financiers et ont, d'autre part, acquis une méthode de développement qui leur est propre (et donc aucunement standardisée) et qui leur semble la meilleure en fonction des outils et des informations disponibles. Ces personnes considèrent en général SDM d'une façon moins favorable et l'appliquent en général de manière plus superficielle. Ils ne sont prêts à changer leurs habitudes de travail qu'à condition de percevoir immédiatement les apports de la nouvelle méthode.

2.3.4. Informations les plus réclamées à propos de la méthode SDM

L'objectif de cette question était de s'assurer si les interviewés disposaient de toutes les informations à propos de SDM ou si, au contraire, ils estimaient manquer de certaines informations.

RESULTATS:

informations	nombre
cours pratiques	4
cours	1
TOTAL	5

- 2 réponses proviennent de développeurs ne connaissant pas SDM
- 3 réponses proviennent de développeurs appliquant la philosophie SDM.

La demande de cours pratiques est assez grande, elle provient en général de personnes ayant assisté à l'une des sessions de présentation de la méthode en eurent des perceptions diverses. Les programmeurs et les analystes regrettent son aspect fort théorique: *"il serait utile d'y développer une petite application"* . Ces deux grades professionnels eurent des difficultés à comprendre la présentation des analyses conceptuelles et fonctionnelles (des phases auxquelles ils ne participent pas). D'une manière générale, le manque d'exemples et le manque de schémas et de figures dans les manuels furent regrettés.
Le groupe méthodes qui dans un premier temps avait effectivement organisé des cours exclusivement théoriques afin de présenter les aspects les plus prioritaires de SDM, organise actuellement des "workshop" SDM où des exemples, des applications réduites sont présentées. La création d'un workshop efficace demande en effet beaucoup de temps: il faut choisir un domaine qui intéresse tout le monde et ensuite construire autour de ce domaine toute une problématique pédagogique. Cette problématique devra être choisie minutieusement car, dans le cas contraire, le workshop risque d'être ennuyeux pour les personnes connaissant un peu le domaine et trop difficile pour les personnes qui ne le connaissent pas du tout.
Signalons pour terminer que le taux de réponse très faible semble moins provenir d'une connaissance imparfaite de la méthode engendrée par des informations trop peu nombreuses que d'un manque d'intérêt de la part des interviewés. La majorité de ceux-ci, lorsqu'ils ne connaissent pas ou mal SDM, estiment que les inconvénients d'une méthode de développement de logiciels dépassent ses avantages. Leur désintérêt quant à SDM s'en trouve alors renforcé.

2.3.5. En quoi SDM change-t-elle la façon de travailler des développeurs?

L' objectif de cette question était de voir dans quelle mesure les développeurs connaissant bien la méthode percevaient l'influence que celle-ci peut exercer sur leur façon de travailler.

RESULTATS:

changements	
meilleure prise en compte de tous les éléments	2
meilleur dialogue avec l'utilisateur	1
meilleure documentation	1
TOTAL	4

Les quatre développeurs qui appliquent de près la méthode SDM ont estimé son impact général sur leur travail. Les trois types d'impact font partie des avantages de l'application de SDM à la CGER que nous avons présentés plus haut.
Une meilleure prise en compte de tous les éléments a été jugée comme influence principale par deux informaticiens.
Comme nous le verrons au point 2.4.1, leurs tâches complexes est un des éléments encourageant l'emploi de la méthode. SDM répond bien à leur besoin de part son aspect checklist exhaustive.
Un meilleur dialogue avec l'utilisateur a été cité par un informaticien. Cet informaticien se plaignait du manque de rigueur des demandes utilisateur. Pour y remédier, il a utilisé SDM pour justifier ses revendications d'une analyse des besoins et d'une étude de l'existant plus conséquentes.
Enfin, un analyste a estimé que l'élaboration d'une meilleure documentation constituait l'influence principale qu'exerce SDM sur sa façon de travailler. d'après lui: " *SDM reprend tous les points à documenter, avant, je ne savais pas toujours quoi mettre dans les documents* ".

2.3.6. Quels sont les domaines pour lesquels les effets de SDM se font le plus ressentir?

Les possibilités suivantes étaient suggérées:
- meilleure définition des tâches
- plus grande facilité de développement
- plus grande rapidité de développement
- plus grande qualité des logiciels

RESULTATS:

effets de SDM	nombre de réponses
qualité des logiciels	4
définition des tâches	2
facilité de développement	0
rapidité de développement	0
TOTAL	6

Interprétation des résultats

Le faible taux de réponse s'explique ici par le fait que seulement quatre développeurs appliquent complètement la méthode et qu'il est difficile pour les autres de se rendre compte de son impact exact.
Les développeurs appliquant complètement SDM ou appliquant sa philosophie estiment que les effets de la méthode se font le plus ressentir dans la qualité des logiciels. Des logiciels ayant été conçus avec le concours des différents spécialistes, prenant en compte tous les aspects, bien analysés et bien documentés seront forcément d'une qualité supérieure.
Le deuxième effet par ordre d'importance concerne la définition des tâches des développeurs. Celles-ci sont en effet précisées par la checklist SDM.
Les effets de la méthode sur la facilité et la rapidité de développement ont été jugés négligeables.
SDM comme nous l'avons vu ne propose pas une méthode de développement "toboggan". Aucune méthode d'ailleurs ne peut remplacer la créativité, le savoir et l'expérience des développeurs.
Ceci explique pourquoi la facilité de développement n'est pas un élément qui a été estimé augmenté par l'application de SDM: le travail reste à effectuer, il est tout au plus beaucoup mieux canalisé (la méthode insiste plus sur le quoi faire que sur le comment faire).
La méthode n'a pas non plus été considérée comme bénéfique pour la rapidité des développements et cela principalement à cause de la lourdeur de la documentation à établir. Des outils bien intégrés pourraient remédier à cet état de fait.

2.4. Facteurs de contingence influençant la perception et l'application de SDM

Après avoir vu quelles étaient les impressions des développeurs vis-à-vis de la méthode et comment celle-ci est appliquée, nous pouvons nous demander quels sont les facteurs qui peuvent influencer la perception et l'application d'une méthode telle que SDM. En s'appuyant sur le modèle de LEAVITT, nous examinerons l'impact des quatre facteurs suivants:

- la tâche à effectuer
- la structure de travail
- la situation personnelle des développeurs
- la technologie

2.4.1. La tâche à effectuer

2.4.1.1. L'influence du grade professionnel et du type de tâche

Dans quelle mesure la tâche à effectuer peut-elle amener à considérer une méthode de développement comme une aide et de là inciter son emploi? Comme nous l'avons déjà souligné auparavant, le propre du département informatique est de fournir des produits complexes et uniques selon les spécifications du client. Cette "fabrication à l'unité" exige de la part des développeurs une innovation constante. L'organisation innovatrice ne peut pas s'appuyer sur une forme de standardisation pour coordonner ses activités. Comme le note H. MINTZBERG (1982) : "*l'adhocratie traite les aptitudes et les connaissances de ses experts comme de simples bases sur lesquelles il est possible d'innover*". En plus de la relative inefficience économique des organisations de type adhocratique, les réactions humaines qui s'y retrouvent reflètent souvent l'ambiguïté des postes de travail mal définis, les relations autoritaires obscures et les lignes de communication souples et inorganisées. Voyons si à la CGER de telles réactions ont été exprimées et quelles pourraient être leur influence sur la perception et l'application de SDM.

Mettons en relation le degré d'application de SDM avec les grades et le type de tâches des interviewés.

degré d'application de SDM	nombre de réponses	grades professionnels	type de tâche
complet	4	informaticiens 3 analystes 1 programmeur 0	développements 3 devpt. et maint. 1
philosophie	5	informaticiens 3 analystes 2 programmeur 0	développements 3 maintenances 2
pas	3	informaticien 1 analyste 1 programmeur 1	développements 2 devpt. et maint. 1
ignorance	5	informaticien 0 analystes 3 programmeurs 2	développements 3 maintenances 1 devpt. et maint. 1
non développeurs	16		
SOUS-TOTAL		informaticiens 7 analystes 7 programmeurs 3	développements 11 maintenances 3 devpt. et maint. 3
TOTAL		11	11

Interprétation des résultats

Les informaticiens suivent proportionnellement mieux SDM que les développeurs d'autres grades professionnels. Certains développeurs soulignent la complexité du travail qu'ils ont à effectuer. Cette complexité découle des produits bancaires/ d'assurances pour lesquels ils développent des applications, des tâches qui leur sont attribuées, des produits de support, des nombreuses interrelations entre personnes.
Voici par exemple la mission et les responsabilités des chefs de projet:

mission:

- mener à bien l'étude et la réalisation de systèmes informatiques en réponse à des besoins de gestion prévus dans le cadre d'un plan informatique des projets à réaliser
- animer une équipe de développement informatique et assurer la gestion hebdomadaire des activités de celle-ci
- être le contact permanent entre les utilisateurs et l'équipe informatique

- participer à l'étude et à la conception fonctionnelle et technique des projets qui lui sont confiés. Mettre en oeuvre la solution retenue

responsabilités:

- apporter un avis à toute demande d'automation pour laquelle il est fait appel à sa compétence
- établir avec l'utilisateur le calendrier des réunions de travail, les préparer et les organiser
- écouter et conseiller l'utilisateur. Noter ses remarques et éventuellement les susciter et déterminer la suite à y réserver
- établir les estimations de coût et de temps nécessaires à la réalisation du projet en accord avec le coordinateur de projet
- étudier et proposer la découpe en activités ainsi que le modèle conceptuel des données
- participation au travail de conception fonctionnelle dans le cadre de la définition du projet
- veiller à la progression du projet conformément au planning et en rendre compte à sa hiérarchie au moyen des outils et des documents prévus à cet effet. Informer mensuellement l'utilisateur de l'avancement du projet.
- installer le logiciel en production
- être l'initiateur et le garant de la documentation spécifique du projet selon la méthode arrêtée
- veiller à la formation des utilisateurs
- assurer la maintenance du projet avec le souci:
    * de l'équilibre des coûts et de l'efficacité
    * des performances
    * de la satisfaction des utilisateurs.

Devant l'ampleur et la diversité des tâches qui incombent aux chefs de projet ainsi qu'aux autres développeurs, les contraintes s'élèvent de toutes parts, les projets sont de plus en plus importants, la complexité des produits croît sans cesse, une certaine standardisation était devenue nécessaire.
Tous ces facteurs favorisent l'emploi de SDM. Au plus l'environnement est complexe, au plus une informatisation réussie sera tributaire de la maîtrise par les développeurs des fonctionnalités à automatiser et du soin apporté aux analyses conceptuelles et fonctionnelles: "*sans maîtrise du domaine pour lequel on travaille, l'informatisation ne sera pas réussie*". SDM insiste beaucoup sur l'étape étude de l'existant: elle permet de mieux faire face aux difficultés liées à l'informatisation dans des environnements complexes et d'amener les développeurs à mieux comprendre ce qu'ils ont à automatiser. Certains informaticiens conscients de l'importance cruciale de cette étape

vont plus loin que les consignes SDM: ils vont chez l'utilisateur pour appréhender comment celui-ci organise son travail, quels sont les flux d'information et quels traitements sont effectués sur ces informations pour en arriver à des propositions de réorganisation de travail afin de rendre l'informatisation plus efficace.
SDM s'avère également un bon gérant des nombreuses interrelations entre développeurs. De part le fait qu'énormément d'aspects doivent être pris en compte lors de développements importants, les personnes de la sécurité, les DBA, les personnes de la production,... doivent être consultées lors d'un développement. SDM rappelle tous les aspects à prendre en compte ainsi que les différents responsables à contacter. Dès que les interrelations deviennent nombreuses, une checklist de tous les aspects devient nécessaire.
Un autre facteur jouant en faveur de l'emploi d'une méthode de développement est la répartition des tâches d'un projet en fonction des grades professionnels.
Les informaticiens s'occupent des aspects conceptuels, fonctionnels et des tests d'intégration, les analystes des aspects techniques et les programmeurs de la programmation et des tests de programmes.
Si SDM est appliquée, les analystes et les programmeurs sont assurés qu'ils pourront se baser sur des analyses soigneusement élaborées par les informaticiens. Les interdépendances entre les travaux de chaque développeur sont telles que la répartition des tâches ne peut être laissée à l'appréciation de chacun: une répartition standardisée a pour avantage de clarifier le rôle de chacun de façon à ce que celui-ci reste stable au fil des projets.
En résumé, nous pouvons affirmer qu'au plus les tâches d'un développeur sont variées et complexes, au plus l'emploi d'une méthode de développement de logiciels devient nécessaire.
Pour terminer, soulignons que le tableau de départ ne permet de tirer aucune conclusion quant au rapport entre l'emploi de SDM et le type de tâches (maintenance, développement).

2.4.1.2. Les tâches à effectuer où l'aide d'une méthode de développement se fait le plus ressentir

Nous prendrons comme outil de mesure le degré de standardisabilité des étapes SDM : il s'agira d'étapes qui demandent peu de créativité et/ou d'étapes qui peuvent clairement être décomposées en une liste exhaustive de phases plus simples. La méthode de développement s'approchera alors d'un algorithme de développement.

Nous nous baserons :

- sur la description des étapes faite dans [groupe méthodes, 1989]
- sur les descriptions que nous en ont faits les interviewés
- sur notre expérience personnelle

ETAPES:

- Déterminer les objectifs et exigences du nouveau système (étape 1.4).
  L'aspect checklist de la méthode constitue une aide efficace pour l'accomplissement de cette tâche. Elle fournit une liste de critères à prendre en compte. L'oubli de certains de ces critères pourrait engendrer des modifications des objectifs du S.I. au cours de son développement et ainsi être la cause d'importants coûts de modification.
- Etude de l'existant (rassembler et analyser les données) (étape 1.3).
  La méthode propose une liste assez détaillée des sous-étapes de l'étude de l'existant. Des développeurs ont émis le souhait que cette liste soit reprise sur mainframe.
- Faire un schéma du système (étape 1.6).
  La construction du modèle conceptuel des données et d'un diagramme HIPO peut être effectuée d'une manière assez automatique.
  Prenons par exemple la liste des étapes de contrôle d'un schéma E/A:
    - élaboration de la liste brute des informations
    - épuration de la liste brute
    - dégagement des types d'entités
    - rattachement de leurs propriétés aux entités
    - définition des relations inter-entités, y rattacher leurs propriétés
    - détermination des cardinalités
    - simplification éventuelle du schéma.
- Déterminer les problèmes de conversion et de mise en production (étape 1.9)
  En 13 sous-étapes, la méthode SDM propose une liste exhaustive des actions à entreprendre et des éléments à prendre en compte pour l'élaboration de cette étape.
- Rédaction des rapports.
  Le contenu des différents rapports à établir tout au long d'un projet est précisément décrit dans la méthode.
- Spécifier les exigences en sécurité et en confidentialité/Conception des mesures de sécurité (étapes 2.7, 3.4).
  SDM présente une liste des divers aspects de sécurité et de confidentialité à prendre en compte et rappelle l'utilité de contacter le groupe sécu-

rité.

- Déterminer les problèmes humains et les solutions (étape 2.8).
  Ici également, une liste des éléments importants est reprise de façon à guider les développeurs lors de l'élaboration de cette étape.
- Codifier les programmes.
  L'application de SDM préconise l'emploi de la programmation structurée de Jackson qui va partiellement standardiser la construction du code.

2.4.1.3. Les tâches à effectuer où l'aide d'une méthode de développement se fait le moins ressentir

Nous prendrons comme outil de mesure le degré de standardisabilité des étapes SDM : il s'agira d'étapes qui demandent beaucoup de créativité et/ou d'étapes qui peuvent difficilement être décomposées en une liste exhaustive de phases plus simples. La méthode ne pourra pas dans ce cas se rapprocher d'un algorithme de développement.

ETAPES:

- Déterminer les diverses solutions possibles (étape 1.8).
  Cette étape doit avoir lieu lors de l'analyse conceptuelle et a pour objectif de proposer plusieurs solutions sur base de facteurs tels que les coûts de développement, la connaissance du domaine,... . L'établissement des diverses alternatives avec leurs avantages et leurs inconvénients est peu standardisable, il va essentiellement dépendre de l'expérience du développeur.
- Elaborer une vue d'ensemble des coûts et des bénéfices (étape 1.10).
  La description de cette étape est:
    - établir les coûts de développement
    - établir les coûts d'utilisation
      * capacité homme
      * coûts de location
      * entretien

  Vu la complexité de cette étape, SDM ne va pas plus en détail. Ici également, le facteur connaissance de l'environnement joue un rôle important.
- Diviser le système en sous-systèmes et les décrire (étape 2.3).
  Les consignes de la méthode sont:
    * diviser le système en sous-systèmes sur base fonctionnelle
    * minimaliser les plages de recouvrement
    * décrire en détail chaque action

SDM reste ici également fort générale et renvoie aux techniques DFD, HIPO et SHAT pour mener à bien cette étape.

- Elaborer les diagrammes et les descriptions des traitements(étape 2.5/6).
  Le premier point de la description est:
  * diviser chaque sous-système en (sous-)fonctions et examiner par quels processus ceux-ci peuvent être réalisés (élaborer éventuellement plusieurs possibilités).

  La même remarque est applicable ici.
- Spécifier les facilités requises en hardware et en software (étape 2.10/12).
  La méthode indique que le résultat de cette étape dépendra:
  * de la nature du traitement
  * du nombre de transactions désirées
  * des temps de réponse désirés
  * de la grandeur des collections de données
  * du langage de programmation
  * des dispositions prises pour la sécurité et pour la confidentialité

  L'expérience et le savoir-faire des développeurs seront d'une grande utilité pour l'élaboration de cette étape vu son caractère peu standardisable.
- Conception de la structure de stockage (étape 3.3).
  Un des points de la description est:
  * chercher l'efficience, chercher la simplicité, éviter la redondance.

  Ces conseils restent généraux et supposent une connaissance approfondie des bases de données.
- Elaborer les descriptions de programmes (étape 3.5/6).
  Un des points de la description est:
  * élaborer les spécifications des programmes (éventuellement en pseudo-code).

  L'élaboration des spécifications est peu standardisable, elle se fera sur base des processus que devra effectuer le futur S.I. et en groupant les traitements similaires.
- Elaboration d'un plan détaillé de programmation et de tests (étape 3.8).
  Les analystes doivent estimer les durées de codage, de compilation et de tests. SDM ne précise pas davantage comment arriver à ces grandeurs.
  L'expérience des analystes jouera ici aussi un rôle primordial.
- Constituer des données de test (étape 4.6).
  Des tests couvrant toutes les tâches et tous les chemins possibles d'un programme sont très difficiles à réaliser. A plusieurs reprises, les développeurs qui s'en occupent nous ont fait part de leurs regrets concernant l'absence de générateur de données de test. Signalons que dans ce cas,

des standards pourront s'imposer par le biais des outils.

Au vu de cette liste non exhaustive, nous pouvons constater que des étapes importantes sont difficilement automatisables et continuent à requérir le savoir faire et l'expérience des professionnels de l'informatique.
Le groupe méthodes, comme nous l'avons déjà vu, va détailler certaines étapes de la méthode afin d'augmenter l'aide apportée.

2.4.2. L'organisation et les outils de travail

2.4.2.1. Quels sont les éléments améliorables dans l'environnement de développement ?

Cette question avait pour objectif de déterminer les principaux éléments de l'environnement de développement qui étaient perçus comme défavorables afin d' analyser par la suite à quel point la méthode SDM pouvait y remédier. Si SDM y remédie efficacement, la présence de ces élément constitue un incitatif à l'emploi de la méthode.

RESULTATS:

		grades			degré d'application de SDM				
	nb	infor.	analyst	progr.	près	philos.	pas	ignore	hors
interface utilisateur plus formelle	5	2	1	2			1	3	1
des outils plus intégrés	4	3	1	0	1	2		1	
une définition des tâches plus détaillée	3	2	1	0		1			2
une adaptation de SDM à chaque poste de travail	3	2	1	0	2	1			
des contacts plus nombreux avec les groupes de support	3	0	1	2				2	1
des outils plus souples des outils plus nombreux	3 2	3	2	0	3	1		1	
des cours moins théoriques	1	0	1	0		1			
des analyses faites par l'informaticien plus élaborées	1	0	0	1				1	
TOTAL	25	12	8	5	6	6	1	8	4

Interprétation des résultats

a. Une interface utilisateurs/informatique plus formelle

La demande d'une interface plus formelle est clairement liée à la non application de la méthode SDM.
Les développeurs ayant émis cette demande ont signalé le fait que des utilisateurs (responsables ou non du projet en cours) expriment leurs plaintes et leurs demandes d'ajout de fonctionnalités aux programmes de manière non formelle et parfois même téléphoniquement. Officiellement toutefois, toutes les demandes utilisateur doivent être justifiées, cataloguées et suivies.
Dans ce cas, une minorité de développeurs n'estiment pas cet état de fait gênant et préfèrent garder leur organisation de travail plutôt que d'introduire des structures de dialogue standardisées. Ces développeurs ne perçoivent pas l'aide que SDM peut apporter à ce problème.
La majorité de développeurs cependant regrettent ce manque de standardisation et veulent remédier à ce manque de formalisme: *"il faut absolument centraliser les demandes utilisateur car cela part dans tous les sens".* Un dialogue utilisateurs-informatique formalisé permet également de résoudre un des problèmes majeurs évoqués par de nombreux développeurs: le manque de connaissances total des utilisateurs quant à la façon de travailler des informaticiens et quant aux problèmes informatiques. Les utilisateurs ne comprennent pas souvent les aspects techniques et les contraintes propres à l'informatique. Ils n'ont pas de vision globale du projet et ne raisonnent pas à long terme. Des réunions explicatives peuvent dans ce cas être fort positives car les utilisateurs se rendront mieux compte des contraintes et des enjeux, ils pourront formuler des demandes plus précises et plus réalistes. Ils comprendront par exemple mieux pourquoi une application nécessitera x semaines de développement alors que la même application (à leurs yeux du moins) peut être réalisée beaucoup plus rapidement sur PC. Une grande majorité d'utilisateurs éprouvent des difficultés à exprimer clairement les spécifications des futures applications: comme ils n'ont pas de vision globale du projet, ils expriment plutôt leurs desiderata en fonction de cas particuliers et non pas en fonction d'un cas général acceptant tous les cas particuliers. Ce manque de précision engendre des malentendus: les développeurs pensent que les utilisateurs ne savent pas ce qu'ils veulent puisque, comme les limites du projet n'ont pas été clairement fixées au départ, ces derniers demandent l'ajout de fonctions supplémentaires en cours de développement. L'importance d'une discussion entre les deux parties est de moins en moins mise en doute. Le chef de projet doit aller voir chez l'utilisateur comment le travail est organisé, quelles sont les informations employées, quels traitements sont appliqués à ces informations. Après cette étude de l'existant, la spécification du projet pourra être entamée. Cette spécification sera facilitée si le représentant des utilisateurs a quelques notions d'informatique: s'il connaît les possibilités et

les contraintes liées aux clés d'accès, s'il comprend la notion de cardinalités dans un schéma E/R... . Dans certains cas, et pour cette raison, des représentants suivent des cours informatiques (SDM, ...).
Conscients du travail des développeurs et des contraintes informatiques, ils deviennent de meilleurs interlocuteurs.
Une interface utilisateurs/informatique peu formalisée constitue donc un puissant incitatif à l'adoption de la méthode SDM (surtout durant la première phase d'un développement).

Lorsque la méthode SDM est utilisée, les rapports entre les utilisateurs et l'informatique sont forts formalisés (étude de l'existant approfondie, limites du projet clairement définies au départ, centralisation et formalisation des demandes de modification,...). L'interface utilisateurs/informatique est alors jugée suffisante par les développeurs.

b. Des outils de développement plus intégrés

Les demandes d'outils plus intégrés proviennent principalement des informaticiens appliquant de près ou appliquant la philosophie de SDM. Comme nous l'avons déjà vu, les tâches des informaticiens sont nombreuses et diverses, il est donc intéressant pour eux de disposer d'outils capables de communiquer entre eux de façon à ce que la récupération d'informations soit maximale. Nous ne pouvons rien conclure quant au rapport entre la demande d'outils plus intégrés et le degré d'application de SDM vu que trois réponses sont issues d'informaticiens développeurs et que plus de 85% d'entre eux appliquent SDM de près ou de loin.
Nous avons par contre vu au point 2.3.3.d que des outils mal intégrés peuvent dissuader l'emploi de la méthode.
La méthode de travail dépend forcément des outils existants, en utilisant les outils, les développeurs sont obligés de suivre la méthode proposée. "*La méthode suivie dépend des outils, on se coule dans le dictionnaire de données, dans les macro-instructions, ...*" .
La méthode SDM n'est pas supportée par un outil intégré, ce qui implique que les interfaces des outils existants ne sont pas compatibles. Le manque de possibilités graphiques du dictionnaire des données n'est pas non plus incitatif à l'emploi de la méthode. Si cet outil est indispensable en tant gestionnaire de liens (quelles sont toutes les routines qui utilisent la routine x ?, ...) il est fort lourd à utiliser pour élaborer le schéma E/R par exemple. Un outil informatique (DOCLIB) a d'ailleurs été créé dans le secteur des assurances pour y pallier. Cet outil sert à documenter les analyses et sa convivialité est largement appréciée. Néanmoins, il ne dispose pas des possibilités graphiques d'EXCELERATOR. Les services support ont mis un tas de macro-ins-

tructions à la disposition des développeurs ex: appel BD, ... Ces macro-instructions teintent l'environnement de développement et ne facilitent pas la vie à un changement d'organisation du travail.
En conclusion, soulignons le lien étroit qui existe entre les outils, l'intégration de ces outils et l'application de la méthode de développement .

Le groupe méthodes et le groupe outils sont en train de travailler à l'élaboration d'interfaces entre outils. L'intégration des outils est un des soucis principaux de ces groupes.

c. Des définitions des tâches plus détaillées

La demande de tâches plus détaillées provient d'abord d'informaticiens puis d'analystes. Cela peut s'expliquer par le fait que les tâches des informaticiens sont complexes et variées et que, si les analystes et les programmeurs reçoivent une guidance plus ou moins importante de l'informaticien responsable du projet, lui, est investi de beaucoup plus de responsabilités.
Est-ce que le besoin d'une plus grande définition des tâches favorise l'application de SDM et, une fois introduite, qu'elle est l'influence de SDM sur cette définition? En général, la définition des tâches des développeurs à la CGER ne va pas jusque dans les moindres détails, une autonomie assez importante est laissée aux développeurs qui peuvent en général choisir la manière dont il vont mener à bien l'objectif qui leur a été assigné. Certains développeurs trouvent cette autonomie gênante et sont donc demandeurs d'une plus grande précision de la définition de leur travail. Ces développeurs ont une appréciation favorable de SDM et essayent de la suivre aussi rigoureusement que possible lorsque le temps de développement le leur permet. Une majorité d'entre eux estiment cependant que SDM ne va pas assez loin dans la précision des tâches car ils doivent aller y pêcher les étapes qui les concernent et, de plus, la zone d'imprécision reste très grande à l'intérieur de chaque étape. Ils demandent une définition des postes de travail: pour chaque poste serait élaborée une liste des tâches à effectuer et, pour chacune d'elles seraient proposés une méthode et des outils de support. D'autres partisans des postes de travail définis de façon très détaillée trouvent SDM suffisante car une zone d'autonomie au sein de chaque étape doit subsister, mais trouvent que la méthode devrait être appliquée beaucoup plus strictement avec des points de contrôle garants de la qualité des analyses beaucoup plus nombreux.
Pour répondre à la question de départ, on constate qu'une définition des tâches peu élaborée laissant une marge de manoeuvre importante aux développeurs a pour conséquence d'engendrer chez certains un sentiment d'insécurité et donc de favoriser l'emploi d'une méthode de développement

telle SDM. Toutefois, l'application de celle-ci à la CGER a voulu justement éviter une guidance trop rigoureuse ne laissant aucune autonomie à ses utilisateurs. Il se peut donc, et les interviews le confirment, que la guidance de SDM n'est pas perçue comme suffisante par certains développeurs.
Une fois appliquée, SDM va à son tour influencer la définition des tâches de chacun en proposant une liste d'étapes à suivre et à respecter lors d'un projet.

d. Adaptation de SDM à chaque poste de travail

Comme nous venons de le voir,SDM devrait, d'après certains développeurs qui l'appliquent de près ou de loin, être plus détaillée et même être adaptée à chaque poste de travail. Comme mentionné plus haut, le groupe méthodes envisage de détailler plus certaines étapes SDM dans un futur proche comme il l'a déjà fait en partie pour les étapes 1.6 (MCD) et 1.9 (normalisation, PALOMA).

e. Contacts plus nombreux avec les groupes de support

Ce sont les développeurs des niveaux les plus bas qui réclament une meilleure collaboration avec les groupes de support. Cette meilleure collaboration permettrait:

- d'améliorer leurs connaissances à propos des outils et des méthodes
- de mieux maîtriser les bases de données.

Le contact avec les groupes de support étant formalisé dans SDM, le sentiment d'une collaboration imparfaite développeurs/support constitue un facteur de contingence favorisant l'emploi de la méthode.
Nous avons également posé la question: comment la collaboration avec les groupes de support pourrait-elle être améliorée ? Les réponses se répartissent comme suit:

améliorations	nombre	d° d'application
plus de réunions	4	pas 1 philosophie 1 ignorent 2
brochure des groupes de support	3	pas 2 philosophie 1
plus de formations	3	pas 1 ignorent 2
formalisation des contacts	1	non dévelop-peurs 1
TOTAL	11	

Au vu des réponses, nous pouvons effectivement remarquer qu'aucune ne provient de développeurs appliquant SDM de près. Des réunions plus nombreuses, des contacts plus réguliers est la principale proposition des développeurs ne suivant pas de près la méthode.

f. Outils plus souples/outils plus nombreux

Comme pour la demande d'outils plus intégrés, les réponses proviennent d'informaticiens et d'analystes, pour les mêmes raisons.

2.4.2.2. Quels sont les domaines où une information ou une aide supplémentaire seraient les mieux venues

L'objectif de cette question était de cerner les éléments de l'environnement de développement pour lesquels la circulation de l'information était ressentie comme insuffisant afin d'analyser par la suite l'aide que pouvait y apporter SDM.

RESULTATS:

		grades			degré d'application de SDM				
demandes	nb	infor.	analyst	progr.	près	philos.	pas	ignore	hors
outils de développement	9	2	5	2		3	1	4	1
définition des tâches	6	3	3	0		2	2		2
produits financiers	2	1	1	0		1	1		
TOTAL	17	6	9	2	0	6	4	4	3

a. Les outils de développement

La majorité des développeurs ayant répondu à cette question estiment que la circulation des informations à propos des outils informatiques existants pourrait être améliorée: "*il y a une foule d'outils que les gens ne connaissent pas et qui pourraient les aider*". L'information concernant les outils n'est pas centralisée quelque part, la plupart des développeurs continuent à employer les outils auxquels ils sont habitués même si d'autres plus efficaces existent dans le département. Ceux qui pourtant sont conscients de l'aide que pourraient apporter de nouveaux outils doivent aller chercher l'information où elle se trouve, ce qui engendre une perte de temps considérable. Un autre effet négatif de cette situation est le fait que certaines unités développent des outils qui existent dans d'autres unités ou développent des outils qui pourraient être utiles à d'autres unités, mais puisqu'aucune centralisation n'est effectuée, ces outils ne sortiront pas rapidement de leur lieu de création.
A propos des outils qu'ils utilisent, les développeurs estiment que plus de formations pourrait les aider: "*des outils réagissent de façon bizarre dans certaines conditions, les manuels n'expliquent pas pourquoi*".
Des outils ne sont donc pas toujours employés au maximum de leur capacité car les utilisateurs ne les maîtrisent pas suffisamment. Le dictionnaire de données DATAMANAGER par exemple recouvre entre autres trois domaines de documentation correspondant à trois phases essentielles:

- analyse fonctionnelle: FUDOS

documentation sur la "tâche", "l'objet", "l'élément"

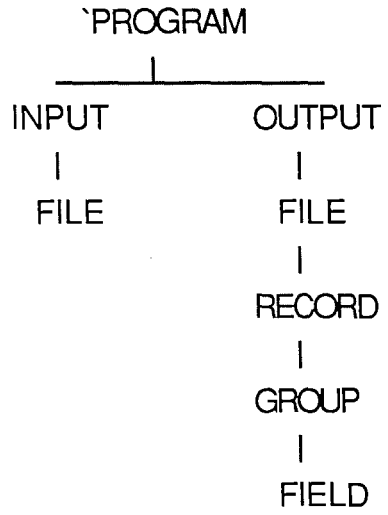
- analyse technique:TEDOS

documentation sur "le programme", "le record", "le field"

- mise en production: PRODOS

documentation sur "le job", "le dataset".
La documentation est organisée de façon hiérarchique.
Exemple:

- un *file* contient des records composés eux-mêmes de *groups* et ceux-ci de *fields*
- un programme fait appel à des routines qui utilisent des *parameters*



Pour chaque membre (program,file, field,...) une série de clauses de description sont prévues
ex:fichier- volume,...
chaque membre est doté d'une clause de commentaire (texte).Le passage en production nécessite une description des programmes telle indiquée ci-contre.
Certains développeurs remplissent uniquement la clause commentaire des program du TEDOS et se servent de DATAMANAGER comme d'un traitement de texte

La cause principale de cette circulation d'information parfois imparfaite est que le département informatique est devenu tellement important en taille qu'il s'avère impossible d'envoyer la documentation à tout le monde. Des brochures existent, des forums sont organisés, mais la circulation de l'information est devenue difficile dans un département de la taille de celui de la CGER. Le service outils par exemple envoie régulièrement des rapports sur le fruit de ses recherches aux différents coordinateurs de projet. A partir de là, l'information circule de manière informelle. Des brochures des groupes méthodes et outils sont également envoyées aux coordinateurs de projet, mais, vu les remarques de certains développeurs, nous pouvons dans ce cas également parler d'une mauvaise circulation de l'information.
La majorité des développeurs ne sont donc en contact avec les groupes de support que lorsqu'un problème surgit. Ils regrettent cette situation et estiment que l'impulsion devrait venir des groupes support: formations aux outils et aux méthodes de développement plus nombreuses, participation plus active au sein des groupes de travail, ... Concernant ce dernier point, il est important de faire remarquer que les groupes de support organisent de telles formations, rédigent des documents explicatifs, mais que les difficultés semblent provenir d'une mauvaise circulation ou d'une circulation informelle de leurs travaux au sein des différents groupes de développement.
La circulation des information sur les outils de développement pourrait être fortement améliorée par des contacts plus nombreux et plus systématiques entre développeurs et membres des groupes de support. Cela exigerait toutefois un gonflement des effectifs des groupes de support..

b. Informations sur la tâche à effectuer, les produits financiers

Ces développeurs regrettent un manque d'information dans leur situation de travail, informations sur le travail lui-même:"*nous sommes parfois amenés à travailler sur des choses que nous ne comprenons pas*" . Les causes principales sont: une mauvaise connaissance des produits pour lesquels les développeurs sont amenés à créer des applications, une vision trop restreinte du produit pour pouvoir maîtriser l'objet même de leur travail et des spécifications et analyses amont insuffisamment développées.
Ensuite, au sein de leur groupe de travail, des développeurs ne savent pas toujours ce que font les autres même si les interactions entre eux sont fort importantes. Certains développeurs regrettent également le manque de feedback: "*je ne sais pas ce qu'il advient de mes analyses*". Ils désirent plus d'informations sur ce qui est fait de leurs travaux, comment ceux-ci sont employés, ...
Un dernier point relevé à propos de la situation de travail est le manque de continuité lorsqu'un responsable change de poste: "*lorsque quelqu'un change de poste, toute son expérience n'est pas transmise et il incombe à son successeur d'aller chercher l'information là où elle se trouve*". Pour ces personnes, une standardisation plus grande et une formalisation des flux d'information et des modes de passage de cette information sont les bienvenues.
Au vu de ce qui précède, pouvons nous établir un lien entre la bonne ou la mauvaise circulation de l'information (information circulant à l'intérieur d'un groupe de travail, entre les divers groupes de développement, entre les groupes de support et les groupes de développement et celles à propos des postes de travail) et l'application de SDM.
Remarquons tout d'abord qu'aucun développeur appliquant la méthode de près n'a formulé de proposition.
SDM répond en effet aux trois types de propositions émises:

- elle favorise la circulation et la centralisation de l'information à propos des outils de développement (étape 1.7: déterminer les outils et les solutions possibles, 3.7: description des programmes standards à utiliser, et également les étapes qui peuvent être accomplies avec le concours des groupes de support)
- elle fournit des précisions à propos de la définition de la tâche (checklist)
- elle met l'accent sur l'importance d'une bonne étude de l'existant et donc sur la connaissance des produits bancaire/d'assurances.

Une circulation de l'information estimée perfectible est donc un facteur de contingence encourageant l'application de la méthode.
Essayons de comprendre pourquoi dans ce cas les développeurs connaissant la méthode SDM et ayant émis le souhait de recevoir plus

d'informations ne l'appliquent pas. D'après eux, les points faibles de SDM sont les suivants:

SDM engendre trop de travail administratif	4
SDM reste trop vague à l'intérieur des étapes	2
SDM n'est pas assez adaptée aux différents types de développement	1

2.4.2.3. Quelles améliorations suggérez-vous concernant la répartition du travail dans les groupes et l'organisation des tâches des développeurs?

Cette question avait pour objectif de déterminer les problèmes de structure de travail pour ensuite pouvoir analyser dans quelle mesure SDM pouvait y remédier.

RESULTATS:

propositions	nb de réponses	degré d'application de SDM
plus de systématisation du groupe	2	pas 1 ignore 1
avoir un informaticien	2	pas 1 ignore 1
TOTAL	4	

Les quatre améliorations émanent de personnes travaillant dans un environnement où le groupe de travail est peu ou pas du tout formalisé.
Un groupe de travail pourrait être défini comme une réunion temporaire de plusieurs personnes impliquées dans la poursuite d'un objectif commun. Si les développements importants sont dotés d'un groupe de travail pour mener à bien leur réalisation (réunions régulières entre chefs de projet, représentants des utilisateurs, analystes, programmeurs, DBA, éventuellement gens de la production, . . .) il n'en va pas de même pour des développements plus restreints, pour les maintenances, pour les ajouts, ...
La formalisation d'une structure de travail semble aller de pair avec l'emploi systématique d'une méthode de développement:

- Lorsqu'aucune structure n'est mise en place pour un développement, les développeurs de tous niveaux se plaignent, comme nous pouvons le voir, d'un manque de guidance, d'un manque de planification de leurs tâches. Le rôle de chacun n'étant pas clairement défini, les analyses amont ne sont en général que peu élaborées et les analystes et pro-

grammeurs doivent se débrouiller avec des études peu poussées. Dans ce contexte, SDM n'est pas appliquée systématiquement: puisque les responsables du projet n'ont pas estimé nécessaire la création d'un groupe de travail , ils ne considèrent pas non plus utile l'application d'une méthode de développement. L'application de SDM devient alors une affaire personnelle: si un développeur désire clarifier son travail, il la suivra, mais ne devra pas attendre que les personnes intervenant en amont et en aval de sa phase en fassent de même (sauf si le chef de projet impose la méthode de développement). La perception de la méthode est dans ce cas liée à la conscience ou non des problèmes liés au manque de structure de travail: les développeurs favorables aux groupes de travail et à plus de formalisation dans les interactions soulèvent les aspects positifs de SDM (meilleure planification, meilleure guidance, ...). Ceux qui, au contraire, estiment qu'un groupe de travail n'est pas indispensable pour les projets dont ils s'occupent et que l'absence de structure de travail précise n'est pas gênante, mettront en lumière les aspects négatifs de la méthode (lourdeur administrative, manque de temps, ...).

- Dans le cas où un groupe de travail est constitué, les responsables de celui-ci sont davantage conscients des problèmes pouvant survenir suite au manque de précision dans la définition des tâches de chacun et suite au manque de formalisation des interactions entre développeurs. Ces développeurs sont par là même plus enclins à conseiller l'emploi de SDM et l'emploiront eux-même.

2.4.3 La situation personnelle des développeurs

La formation, le grade, l'expérience des développeurs peuvent-ils favoriser ou au contraire défavoriser l'utilisation de SDM ?
Rappellons tout d'abord que SDM est une méthode globale guidant tous les grades professionnels et ne s'adressant pas à un grade particulier.
D'après un coordinateur de projet:"*les informaticiens sont des personnes compliquées qui ont besoin de souplesse et d'autonomie".*
Une méthode de développement n'est efficace que si elle est appliquée de façon souple.
Y a-t'il par ailleurs des postes qui demandent plus de formalisme que d'autres?
Il est plus facile de formaliser le travail d'un programmeur (qui consiste essentiellement à codifier les programmes) que le travail d'un informaticien (qui doit s'occuper de l'étude de l'existant, de la spécification et de la description du futur programme, de la division des descriptions en divers sous-systèmes, des tests d'intégration et de toute la gestion du projet).
Les informaticiens sont également les développeurs qui, proportionnellement,

demandent plus de formalisme . Devant la diversité et la complexité de leurs tâches, ils aimeraient plus de guidance "pour s'y retrouver".
Les analystes et les programmeurs par contre estiment, en général, que leurs tâches sont assez claires surtout lorsque les études des informaticiens ont été fort détaillées.
Nous pouvons en conclure que, plus un poste requiert une diversité et une complexité de tâches importantes, plus ses détenteurs expriment le besoin d'un formalisme et d'une guidance dans l'accomplissement de celles-ci.

L'expérience,elle, semble jouer en défaveur de SDM. Lorsqu'un développeur est nouveau, il aura non seulement probablement été initié à une méthode de développement ou à une méthode de programmation structurée, mais il devra également maîtriser le plus rapidement possible l'environnement dans lequel il se trouve: les produits financiers, les outils informatiques, les méthodes de développement, l'organisation du travail, ... L'aspect liste exhaustive de SDM constitue à cet effet un outil précieux et l'on constate que les jeunes développeurs suivent SDM très rigoureusement.
Les développeurs possédant plus d'expérience par contre connaissent mieux l'environnement et font preuve de plus de réticence vis-à-vis de la méthode. Cette réticence s'explique principalement par deux facteurs:

- l'être humain n'aime pas changer d'habitudes. L'application des étapes SDM engendre forcément une modification de leur méthode de travail .

Au fur et à mesure pourtant, les développeurs se sont rendus compte de l'utilité de SDM et beaucoup ont à l'heure actuelle revu leur a priori négatif

- la connaissance de l'environnement étant plus grande, les développeurs ressentent moins la nécessité d'un guidance: "*avec l'expérience, SDM semble lourde*". "*SDM semble lourde, des étapes semblent inutiles car on connait l'environnement*". "*Pourquoi faire à chaque fois une étude de l'existant si on connait les produits ?*".

Les développeurs plus anciens ont derrière eux tout un passé professionnel et ont acquis au cours des années des habitudes de travail qu'ils estiment les meilleures.
Ceux-ci auront donc, au départ du moins, une attitude plus critique à l'égard des méthodes qui sont destinées à standardiser, à réglementer quelque peu leurs activités.
Les habitudes de travail "sclérosent" un peu l'environnement de développement.
L'influence de la formation semble jouer un rôle négligeable quant à la perception de SDM. Que les développeurs aient suivi une formation informatique ou non, leur perception quant à la méthode n'en semble pas influencée.

2.5.Quelles sont les étapes SDM les moins souvent appliquées ?

Nous avons demandé aux développeurs qui n'appliquaient pas la méthode de nous indiquer, d'après les intitulés de SDM, qu'elles étaient les étapes par lesquelles ils passaient toujours et les étapes par lesquelles ils passaient rarement ou jamais.
Voici les résultats:

1. Etude de l'existant et analyse conceptuelle.

L'étape 1.1 (définir le problème et le champ de l'étude) est suivie car chaque demande de développement doit être introduite formellement.
Les étapes 1.2 et 1.3 (rassembler les données sur la situation existante et analyser et évaluer les données) ne sont pas appliquées systématiquement. D'après l'ampleur du projet, le temps disponible et son expérience, le chef de projet décidera s'il est opportun de mener ces analyses. Les informaticiens accordant peu d'importance à ces étapes estiment que leur connaissance de l'existant à informatiser les dispense de les mener.
Les étapes 1.4, 1.5, 1.6 et 1.7 (déterminer les objectifs et exigences du nouveau système, arrêter les points qui restent à résoudre et les hypothèses de base, faire un schéma du système et déterminer les outils et les solutions possibles) sont largement suivies. Il s'agit en effet de la détermination des objectifs du futur S.I. et de l'élaboration du schéma du système.
L'étape 1.8 par contre (évaluer les solutions et opérer une sélection) ne sera pas effectuée si les analyses du projet n'ont pas dégagé plusieurs solutions possibles.
L'étape 1.9 (déterminer les problèmes de conversion et de mise en production et déterminer les critères d'acceptation) est en général suivie contrairement au point 1.10 (élaborer un planning global et une vue d'ensemble des coûts et bénéfices), car les bénéfices liés à l'informatisation sont d'une part difficilement estimables et d'autre part, il faudra souvent poursuivre l'informatisation même si celle-ci ne s'avère pas rentable.

2. L'analyse conceptuelle.

Les étapes 2.1 et 2.2 (spécifier les exigences du système y compris futures et déterminer le cadre dans lequel le nouveau système devra fonctionner) ne sont généralement pas appliquées, car les développeurs estiment que si l'étape 1.4 (déterminer les objectifs et exigences du nouveau système) a été

faite, il n'est plus nécessaire de la détailler ici.
Les points 2.3, 2.4, 2.5/6 (diviser le système en sous-systèmes et les décrire, définir l'input et l'output par sous-système et les interfaces et élaborer les diagrammes et les descriptions des traitements) constituent le noyau central de l'analyse fonctionnelle: division des spécifications initiales en sous-systèmes plus simples à développer et décrire les entrées, les traitements et les sorties de chaque sous-système.
L'étape 2.7 (spécifier les exigences en sécurité et en confidentialité) est bizarrement parfois considérée comme académique
L'étape 2.8 (déterminer les problèmes humains et les solutions) n'est pas souvent prise en compte , les informaticiens estiment qu'elle est du ressort de l'utilisateur. Les problèmes humains de l'informatisation sont donc souvent relégués à l'utilisateur.
Les points 2.9 (y compris PALOMA), 2.10/12 et 2.13 (concevoir la structure logique des données, spécifier les facilités requises en communication de données, en hardware et en software et élaborer un plan pour la poursuite du développement et la mise en production) sont largement suivis. Il s'agit en effet d'étapes indispensables pour la suite: conception de la structure logique des données, détermination des facilités en communication de données, en hardware et en software et enfin, l'élaboration d'un plan détaillé pour la suite du développement.

3. L'analyse technique.

L'étape 3.1(conception des procédures manuelles) n'est pas souvent appliquée, de nombreux analystes ne savent pas en quoi elle consiste.
Les points 3.2, 3.3, 3.4 , et 3.5/6 (conception des formulaires et de tous les input et output de l'ordinateur, conception de la structure de stockage, conception des mesures de sécurité et élaboration des descriptions de programmes et schémas) sont largement suivis. La structure de stockage, les mesures de sécurité et les descriptions des programmes y sont établies. Les étapes ayant rapport avec les bases de données peuvent être faites en collaboration avec les DBA.
L'étape 3.5/6 étant la plus importante de l'analyse technique.
Les points 3.7 et 3.8 (description des programmes standards à utiliser et élaboration d'un plan détaillé de programmation et de tests) ne sont par contre pas toujours réalisés si le projet est d'importance modeste. Les programmes standards et un plan de programmation et de tests ne sont dans ce cas pas jugés utiles: le temps nécessaire à leur élaboration serait trop important par rapport au temps de développement total.

4. programmation.

Les étapes 4.1, 4.2 et 4.3 (description des tâches, définir les exigences en personnel et en environnement de travail et détailler les descriptions de programmes) ne sont pas appliquées lorsque la programmation doit être effectuée rapidement: comme elles n'engendrent pas de lignes de code, elles ne sont pas considérées comme prioritaires.
Les points 4.4 à 4.7 (4.4 codifier les programmes, compilation et correction, constituer des données de tests, tester les programmes) font partie de l'objet même de la programmation.

CHAPITRE 3 PROPOSITIONS D'AMELIORATION

Ce chapitre propose quelques suggestions d'amélioration sur base des interviews menées, sur base de la connaissance de le méthode SDM et de connaissances informatiques et organisationnelles et non pas sur base d'une quelconque expérience en tant que développeur de Systèmes d'Information à la CGER.
Ces suggestions sont donc celles d'un "externe" n'étant pas impliqué dans le processus de développement et ne devant donc pas respecter les contraintes qu'elles soient de type informatique, organisationnel ou psychologique.

Un premier point abordera la formalisation des groupes de travail dans la méthode SDM.
Un deuxième point, en suivant la liste des étapes, proposera des suggestions d'adaptation de certaines d'entre elles pour les développements de maintenance et d'ajouts et pour les développements de "customisation" de progiciels.
Un troisième point comparera un environnement de logiciels intégrés et l'environnement en place pour en déduire ses points forts et ses points faibles.
Un quatrième point suggérera quelques propositions d'amélioration de la circulation de l'information.

3.1. Formalisation des groupes de travail

La méthode SDM, comme nous l'avons vu lors de sa présentation, propose une checklist générale et fort complète des différentes étapes d'un développement de logiciels. Au delà du résumé des étapes SDM, la CGER a terminé il y a peu un document volumineux reprenant pour chaque étape son but, les techniques applicables, la manière de procéder, la documentation à remplir et l'application de l'étape à la CGER (techniques possibles, cours existants, informations disponibles, ...) . Ce document, vu sa relative nouveauté, n'a pas encore été consulté par beaucoup de développeurs. Son utilité en fera probablement dans un futur proche une référence indispensable. Il répond par exemple à la demande maintes fois exprimée d'une plus grande guidance à l'intérieur de chaque activité.
L'activité 1. 2. par exemple: "*Rassembler les données sur la situation existante* y est ainsi décomposée en huit sous-étapes:

- rassembler tous les rapports, documents, formulaires, données sur les procédures, ampleur des tâches, mesures de sécurité
- interviewer les membres du personnel
- contrôler la justesse des données
- rassembler des exemples de toutes les collections de données
- rassembler des exemples de toutes les entrées (origine) et établir le lien entre input et output.
- rassembler des exemples de sorties (destination)
- rassembler des données concernant les erreurs, retards
- noter les souhaits et suggestions formulés

Un projet envisagé est le "babyoutset"qui reprendrait les éléments intervenant dans une analyse conceptuelle et fonctionnelle, le tout devant être disponible sous EXCELERATOR.

Nous avons toutefois remarqué que ce manuel SDM, qui est en quelque sorte l'adaptation d'une méthode théorique à un environnement particulier, n'aborde pas les problèmes de structures de travail. Nous avons précédemment souligné les avantages qui découlent d'une formalisation de ces structures de travail. Certaines étapes décrivant la formation d'un groupe de travail pourraient être ajoutées à la liste des étapes. En fonction de la nature et de l'avancement du projet, la composition du groupe de travail évoluerait. Principalement composé de responsables utilisateurs et de un ou plusieurs informaticiens au départ, il incorporerait au fil du développement des personnes des services supports (DBA, méthode, outils, sécurité, ...) ainsi que des analystes et programmeurs. Le chef de projet établirait ainsi rapidement (à hauteur des étapes 1.3, 1.4) la composition du groupe de travail dans sa première phase. Au fur et à mesure de l'avancement, d'autres étapes lui rappeleraient la nécessité de revoir cette composition. Le planning de développement serait agrémenté des horaires des réunions du groupe de travail ainsi que de leurs ordres du jour.
Cette prise en compte des structures de travail par une méthode de développement aura l'avantage de formaliser l'organisation du travail et d'ainsi mieux préciser le rôle de chacun et de mieux consulter les groupes support. Certaines plaintes exprimées en seront au moins partiellement résolues. Citons par exemple:

- le regret des développeurs quant aux informations trop peu nombreuses émanant des différents groupes de support
- le regret des développeurs à l'égard du manque d'informations et de consignes à l'intérieur de leur groupe de travail. Si le groupe est formalisé, il abordera forcément la répartition des tâches entre les différents développeurs et la délimitation des travaux de chacun (les analystes

pourront commencer leurs tâches sur base de tel et tel documents établis par l'informaticien).
CAPERS JONES (1989) suite à des entretiens avec des programmeurs a interrogé ceux-ci sur ce qu'ils croyaient être les causes des importants taux d'erreurs dans les modules complexes. Les raisons mises en avant pour justifier la complexité du code original étaient, par ordre décroissant d'importance:

- des spécifications ambiguës ou instables
- un manque de temps pour une réalisation soignée
- un manque de pratique du domaine d'application

la conclusion de l'étude est que bon nombre d'erreurs proviennent en fait de spécifications mal faites.(p.137).

- le regret des groupes de support à l'égard des consultations trop peu nombreuses ou trop tardives des développeurs. Cette situation engendre par exemple:
  - des utilisations de base de données non optimales
  - la non utilisation d'outils informatiques intéressants
  - le non respect des capacités machine ou mémoire.

Dans le prolongement de cette suggestion, le département informatique pourrait en arriver petit à petit à une description des postes de travail (par grade professionnel au début). Cette description a été maintes fois réclamée lors des interviews. On disposerait ainsi d'une méthode de développement adaptée à chaque grade professionnel. Les tâches de chacun ne pourront plus alors être source d'imprécision.

3.2. Adaptation de certaines étapes

La grande généralité de SDM fut un autre regret souvent exprimé. Sans avoir la prétention de proposer une méthode adaptée à chaque type de développement, nous allons quand même souligner quelques points qui nous semblent importants pour les projets de type maintenance/ajouts et pour les projets de type customisation de progiciels.
Les maintenances et ajouts apportés à un logiciel déjà existant diffèrent d'un développement ordinaire car l'existant incorpore l'application existante. Nous allons tenter de formuler quelques suggestions qui nous semblent importantes tout en nous rendant compte que, comme nous n'avons pas eu l'occasion de participer à ces types de projet, certaines suggestions seront incomplètes ou peu réalistes dans l'environnement existant. Nous allons pour se faire examiner chaque étape des trois premières phases de la méthode SDM car il semble que les principales difficultés d'adaptation proviennent du-

rant ces phases. Il est important de signaler avant de commencer que les propositions sont faites pour les ajouts de taille moyenne. Elles ne sont pas de mise pour les ajouts d'une dizaine de lignes dans le code source ni pour les ajouts de fonctionnalités importantes qui peuvent être assimilés à des développements ordinaires et qui devront par conséquent suivre de plus près toutes les étapes.

3.2.1. LES MAINTENANCES ET AJOUTS

Une maintenance ne modifie pas les fonctionnalités du logiciel (changement de format de numéros client, changement de base de données...).
Un ajout par contre va rajouter une ou plusieurs fonctionnalités au programme.

Etapes SDM:

1. Analyse conceptuelle

1.1. définir le problème et le champ d'étude

Même si cette étape peut dans certains cas se réduire à un bref entretien avec un responsable utilisateur, il est important de fixer les buts de la maintenance ou de l'ajout. La demande de maintenance ne viendra parfois pas de l'utilisateur, mais du département informatique (performances, BD, ...) le problème devra néanmoins être spécifié correctement.

1.2. rassembler les données sur la situation existante

La situation existante comprend d'une part le programme de base et sa documentation et d'autre part les traitements effectués chez l'utilisateur.
Il faut d'abord explorer le programme existant et en acquérir une assez bonne connaissance. Or, la documentation d'origine est rarement tout à fait adéquate; des réparations ou des modifications ont pu être effectuées sans avoir été répercutées dans la documentation. Une vigilance particulière s'impose donc lors de l'emploi des analyses anciennes.
La situation existante comporte également les procédures (pas toujours automatisées) qui devront être incorporées au logiciel.

1.3. analyser et évaluer les données

Les points suivants ne pourront être omis:

- analyser les erreurs
- éventuellement analyser la qualité et l'efficacité du système actuel.

1.4. déterminer les objectifs et exigences du nouveau système

Les frontières du projet devront être établies ainsi que les exigences qualitatives (ergonomie, sécurité, ...) et quantitatives (capacité, ...) de la maintenance ou de l'ajout.
Les limites du projet et le temps de développement seront également établis.

1.5/6. faire un schéma du système

Trois types d'ajouts ou de maintenance peuvent survenir:

- le nouveau code vient s'ajouter à l'ancien sous forme de modules discrets
- le nouveau code vient s'ajouter à l'ancien en partie sous forme de modules discrets et en partie sous forme de modifications au code de base existant
- le nouveau code vient s'ajouter entièrement sous forme de changements ou de modifications au code existant

En règle générale, les cas 2 et 3 seront plus difficiles à analyser et à réaliser car les constructions internes du système de base devront être examinées plus en détail.

Si le schéma E/R et le graphe de la statique des traitements existent, il faudra les compléter ou les modifier (ajouts ou modifications d'entités, d'associations, de traitements, ...) pour garder une documentation cohérente. Si ceux-ci n'existent pas, le temps disponible ne permettra en général pas d'en élaborer de complets. Nous suggérons dans ce cas, pour le schéma E/R, de reprendre uniquement les entités et les associations en contact direct avec les modifications ou les ajouts qui vont être apportés. Pour les graphes des traitements, seuls les nouveaux traitements ou les traitements à modifier seront décrits.
Ainsi, une analyse partielle sera disponible pour les modifications ultérieures.

1.7. déterminer les outils et les solutions possibles

En général,les ajouts et maintenances ne pourront être réalisés selon différentes solutions possibles. Cette étape pourra alors être négligée.

1.8. évaluer les solutions et opérer une sélection

La même remarque qu'au point 1. 7 est également d'application ici.

1.9. Déterminer les problèmes de conversion et de mise en production

La conversion est l'emploi de données et de programmes déjà existants. Par mise en production il faut comprendre l'adaptation de l'organisation, l'installation de nouvelles machines, la formation du personnel, ...
La conversion devient ici un point primordial: dans le cas des ajouts, il faudra déterminer les problèmes liés à l'existant (code et données), comment intégrer au mieux le projet dans le code de base. Les problèmes de mise en production ne devront en général pas être abordés vu la taille relativement modeste des projets.

1.10. élaborer un planning global et une vue d'ensemble des coûts et bénéfices

Un planning succinct établissant un échéancier pour chaque phase et une estimation des capacités homme requises suffira.

2. Analyse fonctionnelle

2.1. spécifier les exigences du système y compris futures

Si les buts et les objectifs du projet n'ont pas été suffisamment détaillés lors de l'étape 1.4, il faudra les compléter ici. Durant cette étape doivent également être pris en compte les changements intervenus entre l'étape 1.4 et 2.1. Ces changements peuvent être dûs à l'évolution de certaines données, au développement économique, aux nouvelles fonctions désirées, ... De tels changements n'interviendront pas souvent dans le cas qui nous intéresse. Vu le court laps de temps écoulé entre les étapes 1.4 et 2.1 et vu que l'étape 1.4 a été soigneusement réalisée.

2.2. déterminer le cadre dans lequel devra fonctionner le nouveau système

En fonction des exigences (temps de réponse, maintenance, ...) il faudra examiner si les besoins en homme, en capacité machine, en communication de données devront être modifiés. Une modification de ces éléments sera très rare pour les projets d'ampleur modeste.

2.3. diviser le système en sous-systèmes et les décrire

Le but de cette étape est de subdiviser le système initial en parties plus simples qui peuvent être développées séparément. Cette subdivision ne sera pas souvent nécessaire pour les maintenances ni pour les ajouts de taille modeste. La subdivision préconisée à la CGER se base sur les fonctionnalités. Si la maintenance ou l'ajout incorporent plusieurs fonctions clairement distinctes, cette étape pourra toutefois faciliter les tâches en aval.

2.4. définir l'input et l'output par sous-système et les interfaces

La même remarque est d'application ici.

2.5/6. élaborer les diagrammes et les descriptions des traitements

Lorsque les systèmes dégagés au point 2.3 rassemblent plusieurs traitements, ils devront être détaillés ici. Pour chaque traitement un HIPO ou un DFD sera élaboré.

2.7. spécifier les exigences en sécurité et en confidentialité

Une maintenance ou un ajout exigent parfois des normes de sécurité qui n'avaient pas été prévues lors de l'élaboration du programme de base (sécurité de certaines données, procédures de correction, procédures de recouvrement, ...). Ce point nous semble fort important et peut être accéléré en faisant appel au groupe sécurité.

2.8. déterminer les problèmes humains et les solutions

Dans le cas que nous envisagons, de nouveaux problèmes humains ne surgiront que fort rarement. L'influence sur la motivation des personnes, le hardware, les connaissances requises par les salariés, les problèmes sociaux sera négligeable.

2.9. concevoir la structure logique des données

Cette étape commence au point 2.2 et se termine ici. Il s'agit principalement de grouper les données, les identificateurs, les clés d'accès, les chemins d'accès, les vitesses d'accès requises. Cette étape ne va pas toujours concerner la maintenance et les ajouts. Si de nouvelles données seront utilisées, elle devra toutefois être soigneusement menée (nouveaux chemins d'accès, nouvelles exigences de performance, ...).

2.10/12. spécifier les facilités requises en communication de données en hardware et en software

Cette étape ne concernera pas les maintenances et ajouts. Les lignes de communication, les modems, les concentrateurs , les terminaux, les softwares ne devront pas être modifiés suite au projet.

2.13. Elaborer un plan pour la suite du développement et la mise en production

Le planning détaillé est la résultante de cette phase. En fonction de l'ampleur du projet, celui-ci sera plus ou moins élaboré. La conversion et la mise en production y seront de toute façon fortement simplifiées

3. Analyse technique

3.1. conception des procédures manuelles

Les procédures manuelles décrivent les tâches humaines. Elles sont une aide pour la formation du personnel, pour les directives à donner, pour le contrôle quant à la façon de travailler. Cette étape a dû être effectuée lors du développement du programme de base.
Une maintenance n'exigera pas une modification de la manière de travailler des utilisateurs. Un ajout par contre peut changer partiellement leurs procédures de travail. Il sera intéressant dans ce dernier cas d'analyser et de réorganiser si nécessaire les procédures nouvelles concernées.

3.2. conception des formulaires et de tous les input et output de l'ordinateur

Si de nouveaux input ou output doivent être ajoutés, il faudra élaborer la division éventuelle des listes, des écrans de présentation, des *records*,...

3.3. conception de la structure de stockage

Le résultat de cette phase est la détermination de la base de données à utiliser en fonction de critères tels que le temps d'accès désiré, les limites et les possibilités du SGBD, les coûts, les problèmes de sécurité, les problèmes de confidentialité, ...
Dans le cas des ajouts de fonctionnalités, il se peut que de nouvelles données soient utilisées et cette étape de développement sera alors établie avec le concours des DBA.

3.4. conception des mesures de sécurité

Les mesures à mettre en oeuvre pour satisfaire aux conditions de disponibilité et de sûreté spécifiées au point 2.7 sont élaborées durant cette étape. En fonction des résultats de l'étape 2.7, il faudra veiller:

- à exclure l'indisponibilité du système
- à ce que les dégâts occasionnables soient minimaux
- à ce qu'une réparation rapide soit possible.

3.5/6. élaboration des descriptions de programmes et schémas

Cette étape aboutit:

- pour les programmes batch à une première définition des programmes puis en une subdivision en modules
- pour les programmes en on-line en une définition des modules (les différents traitements d'une transaction) et puis en une combinaison de un ou de plusieurs programmes.

Les techniques utilisées sont principalement la structure des diagrammes proposée par Jackson, les tables de décision et le pseudo-code.
Cette étape reste importante dans le cadre des maintenances et des ajouts car son résultat permettra de guider le programmeur de façon standardisée et efficace.

3.7. description des programmes standards à utiliser

Les spécifications des programmes et schémas donnent des indications quant à la possibilité d'utiliser des programmes ou macro-instructions standards pour, par exemple, lister des données, trier des données, fabriquer des données de test, ... Si la maintenance ou l'ajout de fonctionnalités ont à traiter un point pour lequel de tels outils existent, cette étape s'avère fort utile et peut permettre de gagner pas mal de temps de programmation.

3.8. élaboration d'un plan détaillé de programmation et de tests

Un planning fort élaboré ne sera pas nécessaire sauf pour les projets d'une certaine importance. Si toutefois un planning détaillé est requis, il établira les échéanciers, le personnel à affecter et, éventuellement, des informations concernant la manière de tester et la manière d'élaborer des données de test.

En résumé, reprenons dans un tableau les étapes qui doivent être appliquées comme s'il s'agissait d'un développement normal (colonne(1)), celles qui doivent subir des modifications en fonction du projet (colonne(2)) et celles que l'on peut laisser tomber dans une grande majorité de cas (colonne(3)).

ETAPES	1	2	3
1.1: définir le problème et le champ d'étude		x	
1.2: rassembler les données sur la situation existante		x	
1.3: analyser et évaluer les données		x	
1.4: déterminer les objectifs et exigences du nouveau système		x	
1.5/6: faire un schéma du système		x	
1.7: déterminer les outils et les solutions possibles			x
1.8: évaluer les solutions et opérer une sélection			x
1.9: déterminer les problèmes de conversion et de mise en production		x	
1.10: élaborer un planning global et une vue d'ensemble des coûts et bénéfices		x	
2.1: spécifier les exigences du système y compris futures		x	
2.2: déterminer le cadre dans lequel devra fonctionner le nouveau système			x
2.3: diviser le système en sous-systèmes et les décrire		x	

2.4: définir l'input et l'output par sous-système et les interfaces		x	
2.5/6: élaborer les diagrammes et les descriptions des traitements		x	
2.7: spécifier les exigences en sécurité et en confidentialité	x		
2.8: déterminer les problèmes humains et les solutions			x
2.9: concevoir la structure logique des données		x	
2.10/12: spécifier les facilités requises en communication de données, en hardware et en software			x
2.13: élaborer un plan pour la suite du développement et la mise en production		x	
3.1: conception des procédures manuelles		x	
3.2: conception des formulaires et de tous les input et output de l'ordinateur		x	
3.3: conception des mesures de sécurité	x		
3.5/6: élaboration des descriptions de programmes et schémas	x		
3.7: description des programmes standards à utiliser		x	
3.8. élaboration d'un plan détaillé de programmation et de tests		x	

D'après ce qui précède, nous pouvons constater que si certaines étapes des trois premières phases d'un développement de système d'information doivent être suivies de façon très détaillée et si d'autres peuvent être oubliées, de nombreuses étapes devront selon les cas soit être menées de façon détaillée soit être passées.
Nous pouvons en conclure qu'une méthode de développement pour les maintenances et ajouts devra elle aussi garder un caractère général afin de tenir compte de la grande diversité des cas possibles.

3.2.2. Les adaptations de progiciels

Nous allons maintenant essayer de formuler quelques suggestions concernant la customisation d'un progiciel. Lorsque le département estime plus avantageux (d'après différents critères) d'acheter un logiciel sur le marché, il faudra le soumettre par la suite à une adaptation à l'environnement CGER. Cette adaptation peut porter sur différents facteurs tels que l'augmentation de la charge maximale tolérée, l'adaptation en fonction des mesures de sécurité requises, ...
Nous envisagerons le cas des progiciels ayant trait à une activité de la banque (factoring, assurance, comptabilité, ...) et non des progiciels outils (SGBD, générateurs de données de test, outil d'aide au développement, ...).

1. Etude de l'existant et analyse conceptuelle

les points 1.1 à 1.8 devront être élaborés comme pour des développements ordinaires: définition du problème, rassemblement des données sur la situation existante, analyse et évaluation de ces données, détermination des objectifs, élaboration d'un schéma du système, détermination des outils et des solutions possibles et évaluation des solutions suivie d'une sélection. Durant les deux dernières étapes, le chef de projet envisagera les diverses solutions possibles: développement interne, développement par une firme extérieure, développement en collaboration avec une firme extérieure, achat d'un progiciel. Il établira ensuite un rapport destiné à la direction informatique qui reprendra les avantages et les inconvénients de chaque solution en fonction de divers critères: rapidité du développement, coût, qualité exigée, possibilités d'extension, ...

Le comité directeur sera alors amené à porter son choix sur base de ce rapport.

Il peut toutefois se produire que, lorsqu'un produit financier doit être lancé très rapidement, le département informatique décide d'acheter un progiciel sans passer par toutes les étapes citées. Une étude de l'existant réduite devra alors établir les modifications à apporter.

Un autre type de cas peut également se produire: lorsque la CGER décide de se lancer dans une nouvelle activité, il n'y a pas d'existant. Seuls les objectifs et les différentes solutions possibles pourront être établis.

Les étapes 1.1 à 1.8 établies à ce point ont pris en compte l'existant en place (données, traitements,...). A cet existant va venir s'ajouter le progiciel. Les fonctions offertes par le progiciel devront être identiques à celles décelées lors de l'étude de l'existant. Certaines fonctionnalités du progiciel devront donc être modifiées et certaines fonctionnalités devront lui être ajoutées.

Une étape SDM serait la bienvenue à ce stade décrivant comment "prendre en main" le progiciel et comment analyser ses traitements et sa structure.

Une deuxième étude de l'existant portant sur le progiciel sera alors effectuée:

- rassemblement des données sur la situation existante
- analyse de ces données
- élaboration d'un schéma du système (pour les traitements).
  Ce point sera aussi détaillé que possible, un DFD et une description des traitements seront dans la mesure du possible élaborés dans le but de faciliter l'adaptation et d'accélérer les maintenances et ajouts futurs
- comparer les objectifs et exigences du nouveau système (fonctionnalités, sécurité, capacités,...) avec ce que propose le progiciel
- définir les modalités de l'adaptation.

La durée de cette étape est très difficilement estimable car elle dépendra de la

complexité et de la structure du progiciel. Au terme de l'étape, pourront être comparées les fonctionnalités proposées par le progiciel et celles à informatiser. Le schéma E/R de la première étude de l'existant et le schéma des traitements (à modifier) de la deuxième étude de l'existant serviront de base pour la suite du développement.
Au fond, dans le cas d'un progiciel, l'aspect créatif des analyses devient un aspect de contrôle.

1.9. déterminer les problèmes de conversion et de mise en production

L'impact du progiciel sur les programmes et sur les données existantes sera établi: que peut-on réemployer et comment ? De nouvelles machines devront-elles être mises en place, le personnel devra-t-il suivre une formation complémentaire ?

1.10. élaborer un planning global et une vue d'ensemble des coûts et bénéfices

Un planning global des modifications à apporter sera établi, le calcul des bénéfices de l'adaptation ne sera par contre pas souvent nécessaire.

2. Analyse fonctionnelle

2.1. spécifier les exigences du système y compris futures

Cette étape SDM peut-être appliquée normalement: le contrôle des buts du projet, les développements économiques, la croissance de certaines données, ... doivent être prises en compte.

2.2. déterminer le cadre dans lequel le nouveau système devra fonctionner

La structure organisationnelle à mettre en place, la manière de travailler, les moyens à mettre en oeuvre (communication de données, machines, logiciels, ...), les aspects de sécurité sont à envisager comme pour un développement interne.

2.3. diviser le système en sous-systèmes et les décrire

Le but de cette étape est de subdiviser les modifications à apporter au progiciel en sous-systèmes plus simples et développables indépendamment.Une division fonctionnelle peut être appliquée.

2.4. définir l'input et l'output par sous-système et les interfaces

Un HIPO ou un DFD sera établi par sous-système.

2.5/6. élaborer les diagrammes et les descriptions des traitements

Chaque sous-système peut-être décomposé en un nombre de traitements élémentaires qui seront décrits durant cette étape (input, output, but, ...).

2.7. spécifier les exigences en sécurité et en confidentialité

En fonction des exigences en sécurité décrites au point 1.4, seront faites des propositions pour les mesures de sécurité (non garanties par le progiciel) à prendre en compte. L'ajout de procédures de sécurité est d'ailleurs une des préoccupations majeures de la customisation.

Les étapes 2.8 à 2.13: déterminer les problèmes humains et les solutions, concevoir la structure logique des données et spécifier les facilités requises en communication de données, en hardware et en software, élaborer un plan pour la poursuite du développement doivent être menées de la même manière que pour un développement interne.

3. Analyse technique

Les étapes 3.1 à 3.3 conception des procédures manuelles, conception des formulaires et de tous les inputs et output de l'ordinateur pourront être élaborées normalement.

3.4. conception des mesures de sécurité

Les buts concernant la sécurité décrits au point 2.7 et à ajouter au progiciel seront décrits de façon détaillée durant cette étape: mesures concernant la protection du programme et des données manipulées, réalisation

des procédures de contrôle et de correction, élaboration d'un plan d'urgence, mesures concernant la confidentialité des données, ...

3.5/6. élaboration des descriptions de programmes et schémas

Les différentes modifications qui devront être apportées au progiciel sont décrites en détail durant cette étape. Le but est de parvenir à une description telle que les programmeurs puissent la traduire en un langage de programmation sans rencontrer d'importantes difficultés.

3.7. description des programmes standards à exécuter

Cette étape devra être élaborée dans le cas que nous envisagons puisque des programmes standards pourront venir faciliter les modifications à apporter au progiciel.

3.8. élaboration d'un plan détaillé de programmation et de tests

un planning complet de la suite du développement précisera les échéances, les attributions de chaque développeur, ...

Mettons dans un tableau les étapes qui doivent être appliquées comme s'il s'agissait d'un développement normal (colonne(1)), celles qui doivent subir des modifications en fonction du projet (colonne(2)) et celles que l'on peut laisser tomber dans une grande majorité de cas (colonne(3)).

ETAPES	1	2	3
1.1: définir le problème et le champ d'étude	x		
1.2: rassembler les données sur la situation existante	x		
1.3: analyser et évaluer les données	x		
1.4: déterminer les objectifs et exigences du nouveau système	x		
1.5/6: faire un schéma du système	x		
1.7: déterminer les outils et les solutions possibles	x		
1.8: évaluer les solutions et opérer une sélection	x		
étude de l'existant portant sur le logiciel			
1.9: déterminer les problèmes de conversion et de mise en production	x		
1.10: élaborer un planning global et une vue d'ensemble des coûts et bénéfices		x	

2.1: spécifier les exigences du système y compris futures	x		
2.2: déterminer le cadre dans lequel devra fonctionner le nouveau système	x		
2.3: diviser le système en sous-systèmes et les décrire	x		
2.4: définir l'input et l'output par sous-système et les interfaces	x		
2.5/6: élaborer les diagrammes et les descriptions des traitements	x		
2.7: spécifier les exigences en sécurité et en confidentialité	x		
2.8: déterminer les problèmes humains et les solutions	x		
2.9: concevoir la structure logique des données	x		
2.10/12: spécifier les facilités requises en communication de données, en hardware et en software	x		
2.13: élaborer un plan pour la suite du développement et la mise en production	x		
3.1: conception des procédures manuelles	x		
3.2: conception des formulaires et de tous les input et output de l'ordinateur	x		
3.3: conception des mesures de sécurité	x		
3.5/6: élaboration des descriptions de programmes et schémas	x		
3.7: description des programmes standards à utiliser		x	
3.8. élaboration d'un plan détaillé de programmation et de tests	x		

L'existant dans le cas de l'adaptation d'un progiciel requiert comme nous l'avons vu une double étude: celle des procédures en place et celle du produit acheté.
Ces deux étapes peuvent être effectuées en parallèle par des équipes différentes et seront comparées une fois terminées. Rappelons également la planification difficile de l'étude du progiciel.

3.3. Les outils de support

Une troisième difficulté très souvent citée est celle du manque de temps pour développer des applications documentées. Nous avons pu constater qu'une plus grande rapidité de développement peut être obtenue grâce à des outils de support intégrés.
La même information introduite une seule fois pourra être récupérée par divers outils durant le cycle de développement.
Nous pouvons nous demander quelle est l'influence des outils de développement et de l'environnement en général sur la rapidité de d'élaboration d'un logiciel.

3.3.1. L'influence des outils de support sur le temps de développement

Il nous semble intéressant de mentionner à titre introductif une étude qui a été menée par un groupe de l'IEEE travaillant sur des problèmes de productivité (1981). Le système étudié est la création d'un nouveau compilateur ADA comprenant environ 10000 instructions de code source en langage C. Les programmeurs et analystes des différents groupes connaissent bien les études de compilateurs et la définition du nouveau compilateur ADA. Ces compilateurs sont destinés à être vendus (d'où un effort de documentation externe plus important que pour un produit interne). Les exigences du point de vue de la spécification, de la conception , des analyses et de la qualité finale sont identiques pour les différents groupes.
Le logiciel a été développé par trois groupes de compétence équivalente, mais dans des environnements complètement différents.
Nous retiendrons ici les deux premiers:

- le meilleur environnement comprend:

  * un support intégré texte/graphique pour les spécifications, la conception et la documentation
  * une bonne bibliothèque technique et un accès facile aux informations
  * un terminal installé sur le bureau de chaque programmeur ou analyste qui peuvent sur demande obtenir un terminal supplémentaire à emporter chez eux
  * un stock complet d'outils de références croisées, de recherche de bogues et de jeux d'essais
  * beaucoup de petites salles de réunion pour les revues et les inspections
  * un système automatisé de comptes rendus de défauts et de suivi de ces défauts

- l'environnement moyen comprend:

  * du traitement de texte, mais pas de graphiques automatisés
  * petite bibliothèque technique assez dispersée
  * un terminal pour trois programmeurs ou analystes
  * quelques outils de recherche de défauts, mais la bibliothèque de jeux d'essais est manuelle
  * une ou deux salles de conférence utilisables pour les réunions
  * un système manuel d'enregistrement des défauts

Les durées de développement en hommes-mois enregistrés se présentent comme suit:

	meilleur environnement	environnement moyen
SPECIFICATIONS	42	42
CONCEPTION	53	63
DOCUMENTATION INTERNE	122	366
DOCUMENTATION EXTERNE	113	338
CODAGE	158	188
TESTS	85	95
REPARATION DES DEFAUTS	85	147
TOTAL	658	1239

figure 3.1: influence des outils et de l'environnement sur la productivité du logiciel.

L'élément le plus défavorable dans l'environnement moyen est sans doute le manque d'assistance documentaire convenable. La documentation prend, si des outils ne sont pas présents pour la faciliter, un temps considérable et souvent prohibitif lorsqu'on la compare à la charge de travail des autres postes. Citons les différentes catégories de documentation utiles lors du développement d'une application:

- documents prévisionnels et documents reprenant les techniques employées (établis lors des étapes 1.7, 1.10, 2.10/12, 2.13 et 3.8 de SDM)
- documents de contrôle ou comptes rendus périodiques des résultats réalisés (établis lors des étapes 2.14: *comparer les différences de résultat par rapport au planning et mentionner les causes*, 3.9: *comparer les résultats de l'analyse fonctionnelle et de l'analyse technique (planning, coûts/bénéfices) et expliquer les différences*, 4.8: *élaborer un résumé des problèmes rencontrés, des changements internes et de leurs effets sur le planning et les coûts)*
- documents de gestion du projet: échéanciers, allocation du personnel, comptabilisation du projet élaborés par le chef de projet

- documents techniques qui comprennent les besoins, les objectifs et les spécifications fonctionnelles du futur S.I. (établis lors des étapes 2.1, 2.2, 2.7, 2.8 et 2.10/12)
- documents explicatifs destinés aux utilisateurs (guides d'utilisation, matériel pédagogique, ...) (établis lors de l'étape 6.2: *établir un schéma de formation et compléter les cours*)
- correspondance, notes internes relatives au projet, qui ont été échangées entre les membres de l'équipe.

S'il est dangereux de tirer des conclusions d'un seul exemple, nous pouvons toutefois en déduire la quantité de temps énorme absorbée par la documentation et le besoin d'outils souples et intégrés pour alléger au maximum cette charge. La documentation est en tout cas une activité que l'on ne peut laisser hors planning.
Lors des interviews que nous avons menées, les développeurs ont par exemple regretté la manque d'interfaces entre le logiciel SUPERPROJECT[1] et d'autres outils, l'absence d'un outil informatisé qui guiderait le concepteur dans l'élaboration de l'étude de l'existant, le manque d'outils destinés à l'échange d'informations au sein d'une équipe ou entre différentes équipes et les différents autres services, la difficulté d'extraction des données du dictionnaire DATAMANAGER pour les incorporer dans des documents, ...
Nous allons présenter ci-dessous un environnement de logiciels très conviviaux aidant les concepteurs lors des premières phases de développement et nous effectuerons ensuite une comparaison avec l'environnement en place à la CGER afin de déceler les points forts et les points faibles de plusieurs de ses outils.

3.3.2. un environnement de logiciels intégrés

Le but de ces outils logiciels est de saisir les éléments des différentes analyses et d'introduire toutes les modifications souhaitées. Les concepteurs désirent, pour la facilité de réalisation de ces analyses, la plus grande souplesse possible dans l'acquisition. Cela suppose que celles-ci acceptent d'être momentanément incomplètes ou incohérentes.
Afin de concrétiser notre description, nous allons nous baser sur un outil existant: l'environnement DSL.
Cette description est extraite de [BODART et PIGNEUR (1989, pp.161-195)].

[1] logiciel destiné à établir un planning de développement

a. DSL/SPEC.

DSL/SPEC est un environnement logiciel d'aide à la conception destiné à assister une équipe d'analystes dans l'élaboration initiale et dans le suivi des spécifications, exprimées à l'aide du langage DSL(Dynamic Specifation Language) .
Le langage DSL permet de décrire les spécifications fonctionnelles d'un système d'information à l'aide:

- de spécifications d'un schéma de données
- d'une décomposition des traitements (application/ phase/ fonction)
- d'un modèle de la statique des traitements
- d'un modèle de la dynamique des traitements
- d'un modèle descriptif des ressources disponibles
- d'un diagramme de flux.

DSL/SPEC regroupe un ensemble intégré d'utilitaires ou outils logiciels pour:

- acquérir et mémoriser les spécifications au fur et à mesure de leur élaboration
- sélectionner et valider ces spécifications
- extraire et restituer tout ou partie des spécifications sous différentes formes

Cet ensemble d'outils permet entre autre:

- d'assister le concepteur, il lui sert d'assistant dans la partie administrative de son travail ou, dans certains cas, il peut servir de guide dans l'application d'une démarche.
- de contrôler et de valider les spécifications mémorisées
- de restituer l'ensemble ou une partie de ces spécifications pour les utiliser dans des rapports à remettre aux différentes personnes concernées par le projet
- d'adopter cet environnement logiciel afin de permettre une modification du modèle de conception adopté et l'adjonction de nouveaux utilitaires.

Exemple de spécification DSL d'un schéma de données dans sa forme graphique:

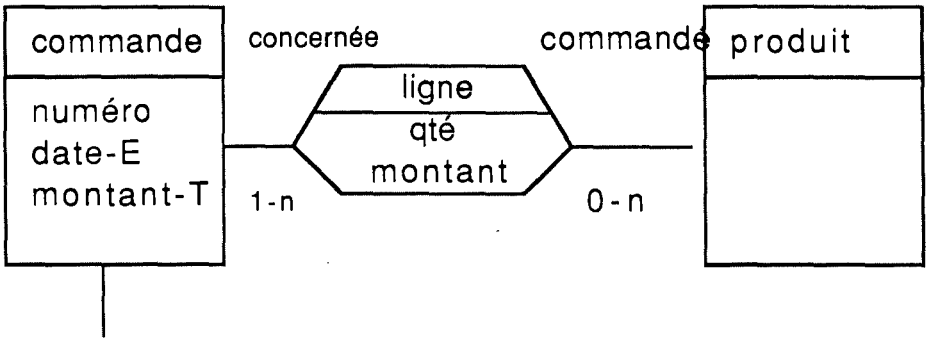


figure 3.2: Spécifications DSL d'un schéma de données.

Texte DSL correspondant à la figure 3.2:

une entité COMMANDE
  par définition:
    représente un ordre de commande passé par un client auprès de la société pour un ou plusieurs articles repris au catalogue
  est caractérisé par:
    un NUMERO
    une DATE d'ENREGISTREMENT
    un MONTANT TOTAL
  joue le rôle
    de COMMANDE CONCERNEE dans une ou plusieurs LIGNE(S)
  est identifié par
    son NUMERO

une association LIGNE
  par définition
    représente la provenance de livraison d'un article commandé
    relie une COMMANDE-CONCERNEE et un PRODUIT-COMMANDE
  est caractérisé par
    une QUANTITE et un MONTANT

Cet environnement de spécifications:

1. S'articule autour du dictionnaire des spécifications, mémoire du processus de conception qui peut-être apparenté à DATAMANAGER.

2. S'adapte à un modèle de spécifications particulier grâce à un méta-dictionnaire du modèle, qui définit la représentation E/A du modèle adapté, en l'occurrence DSL
3. Intègre un ensemble d'utilitaires logiciels d'acquisition, de validation et de restitution des spécifications
4. Offre un gestionnaire du dialogue pour faciliter les interactions avec les concepteurs.

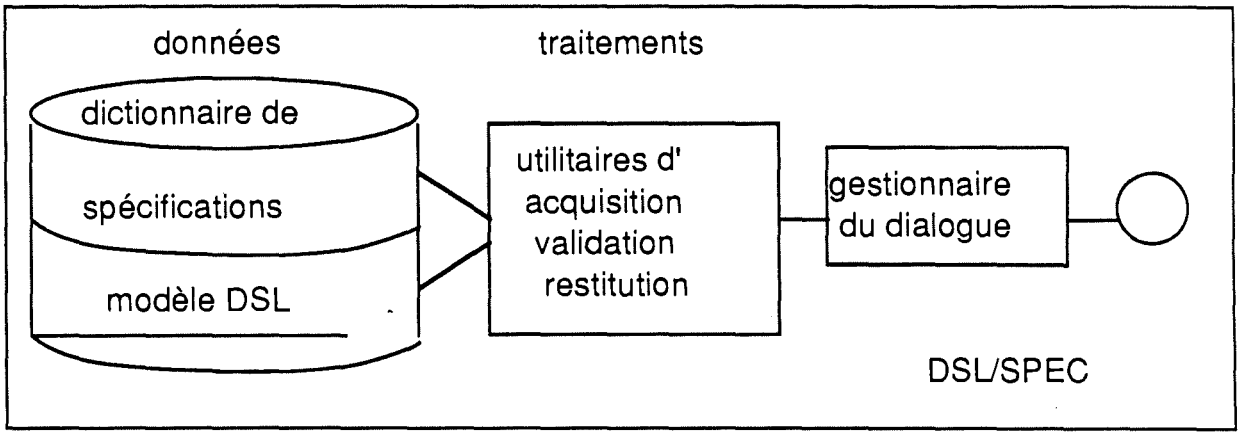


figure 3.3: l'environnement DSL/SPEC.

Le dictionnaire des spécifications.

Comme le dictionnaire de données DMR, le dictionnaire des spécifications constitue le noyau central des développements de logiciels ainsi que des outils de support. Il regroupe l'ensemble des spécifications de système d'information. Les spécifications y sont emmagasinées et structurées selon le schéma E/A.

Le méta-dictionnaire du système.

Le méta-dictionnaire se base sur le fait que la représentation E/A est retenue pour structurer toutes les spécifications dans le dictionnaire. Il renferme la définition du modèle adopté et tous les utilitaires accédant à ce méta-dictionnaire pour s'adapter au modèle souhaité. Il pilote donc les utilitaires et admet la représentation E/A, la syntaxe autorisée par la forme textuelle de DSL, les icônes pour sa forme graphique et les règles de validation exprimées en DSL/QS

les utilitaires d'acquisition et de mémorisation.

Leur objectif est de saisir et d'enregistrer dans le dictionnaire les spécifications DSL au fur et à mesure de leur élaboration.
L'acquisition est souple par deux aspects:

- les spécifications introduites peuvent momentanément être incohérentes ou incomplètes
- grâce à un éditeur graphique, le concepteur pourra selon son choix choisir la forme de l'introduction des données. Les types d'entités, types d'association, processus, points de synchronisation, ... pourront être directement sélectionnés par des icônes.

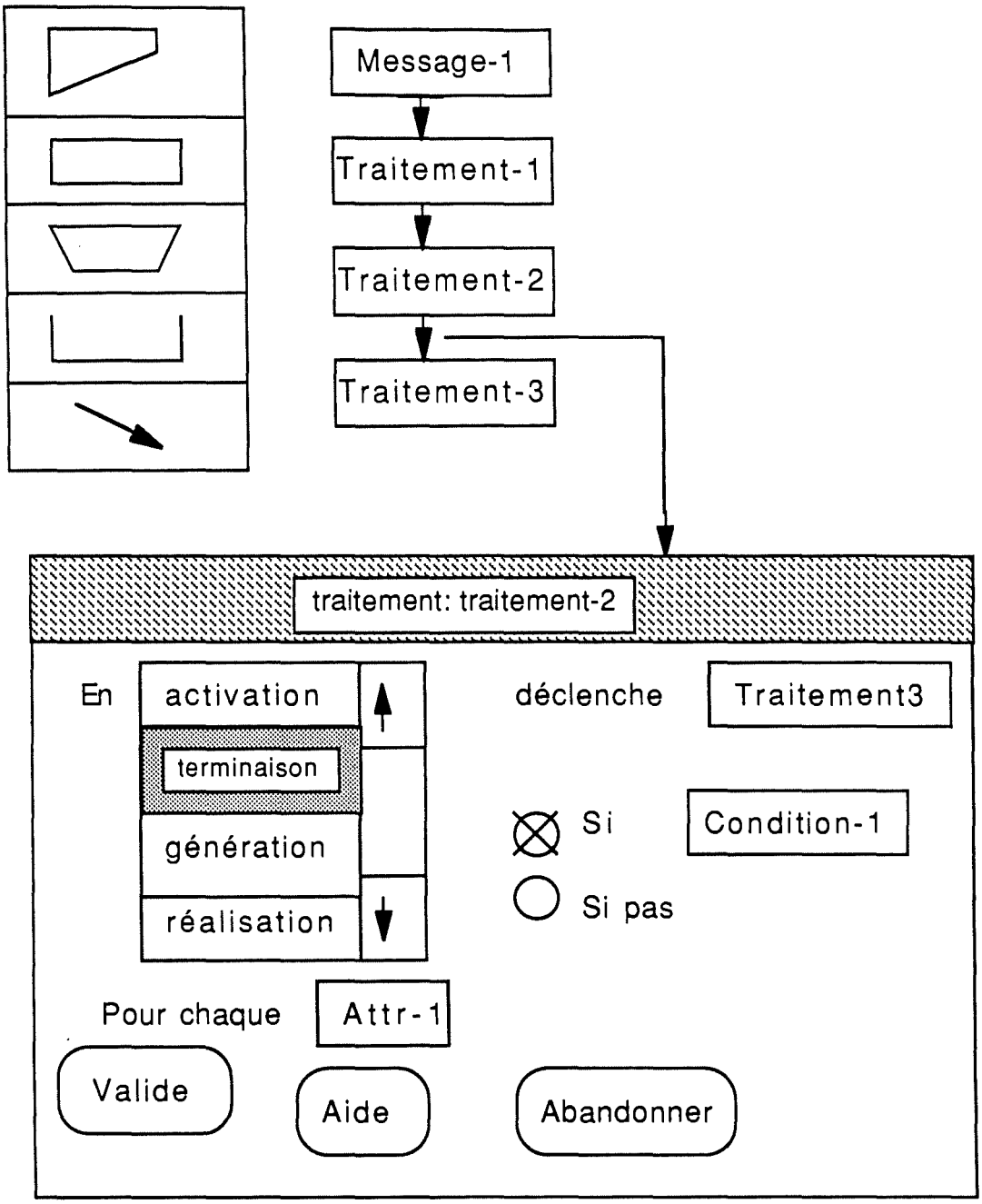


figure 3.4: Description graphique d'un traitement

Les utilitaires de sélection et de validation.

Un langage d'interrogation (DSL/QS) permet de retrouver dans le dictionnaire des ensembles d'entités DSL qui satisfont à certains critères de recherche. La recherche peut s'effectuer sur:

- le type des entités

- leur nom et leurs synonymes éventuels
- leurs attributs et leurs valeurs
- leur rôle dans certaines associations

exemples:

quels sont les traitements qui reçoivent le message "BON DE COMMANDE" ?

quelles sont les "phases" et les "fonctions" décrites dans le dictionnaire ?

Des utilitaires de validation permettent de vérifier si les données introduites respectent les règles précisées par les développeurs. Si une règle prévoit que "tout traitement reçoit au moins un message", tous les traitements sont examinés et ceux qui ne la respectent pas seront sélectionnés.

Les utilitaires d'extraction et de restitution.

Des rapports destinés aux différentes parties ayant un intérêt avec le projet en cours devront être régulièrement élaborés. Tout ou partie des spécifications introduites dans le dictionnaire devra être incorporée dans certains documents.
L'extraction des spécifications se fait en fonction du choix des demandeurs qui doivent pouvoir sélectionner les composantes de la spécification qui les intéressent.
La restitution sous forme d'un texte consiste à restituer des spécifications qui concernent les entités sélectionnées sous la forme d'un texte DSL.
La restitution sous forme graphique est également possible: un utilitaire va prendre automatiquement les schémas graphiques à partir du dictionnaire des spécifications que celles-ci aient été initialement acquises sous forme d'un texte ou sous forme d'un graphique. Les quatre types de schémas suivants peuvent être restitués:

schéma entité/association
schéma de la décomposition des traitements
schéma de la dynamique des traitements
diagramme de flux

La différence principale entre le dictionnaire de données DATAMANAGER et l'environnement de spécification offert par DSL/SPEC réside dans les possibilités graphiques. Celles-ci sont inexistantes pour DMR. L'acquisition, la modification et la restitution des données en DSL/SPEC peut se faire sous la forme choisie par les concepteurs et particulièrement sous forme graphique. L'acquisition et la modification sous forme graphique permet notamment des améliorations progressives du schéma E/A, des diagrammes des flux, du schéma de la dynamique des traitements.

b. DSL/SIM

L'outil d'évaluation DSL/SIM est un logiciel de production de mesures statistiques du comportement dans en "exécutant" les spécifications fonctionnelles du comportement d'un S.I. .
Cet outil comporte trois éléments:

1. le générateur automatique: un programme de simulation qui correspond à une version exécutable des spécifications du comportement enregistrées dans le dictionnaire de l'environnement de spécifications.
2. l'exécution de ce programme de simulation qui engendre la création d'une base de données statistiques
3. l'exploitation de cette base de données statistique, description du comportement simulé pour élaborer des rapports d'analyse.

A partir des spécifications enregistrées dans le dictionnaire, l'outil DSL/SIM va donc générer un programme automatiquement. Il va pour ce faire se baser sur la dynamique des traitements, la charge estimée et les moyens ou ressources requises. L'utilisateur peut ensuite préciser la durée de simulation du programme généré. Elle peut être interrompue afin d'obtenir une trace d'exécution chronologique. Cette simulation va engendrer des résultats statistiques qui vont permettre de comparer les performances obtenues avec les performances attendues et également permettre de localiser la cause des comportements anormaux ou insatisfaisants.
Les mesures exploitées dans les résultats statistiques portent:

- soit sur des nombres de changements d'état caractéristiques du comportement des objets considérés (ex: le nombre de déclenchements de processus d'un certain type)
- soit sur des durées de certains états caractéristiques par lesquels évoluent les objets considérés (ex: la durée d'attente d'un processus)
- soit sur des volumes ou nombres d'objets dans certains états caractéristiques à un moment donné (ex: le nombre de processus en attente ou la capacité disponible d'une ressource à l'instant t)

Les aspects de simulation détectables par les résultats statistiques sont:

- les divergences entre performance attendue et obtenue
- les retards anormaux dans le déclenchement des processus
- les attentes anormales de processus déclenchés
- les ressources critiques

DSL/SIM permet donc d'appréhender le caractère réalisable des spécifications fonctionnelles introduites dans le dictionnaire du point de vue des performances. Cet outil ne permet par contre pas de vérifier si les spécifications

fourniront des résultats conformes à ceux souhaités par les utilisateurs.

c. DSL/PROTO.

L'outil de prototypage DSL/PROTO répond à ce besoin. Il permet une expérimentation progressive des spécifications introduites: exécution à différents niveaux: "fonction", "phase", "application".
L'exécution des spécifications permet de ne pas devoir attendre l'implémentation finale du projet et de se rendre très rapidement compte des divergences entre spécifications et souhaits des utilisateurs.
DSL/PROTO gère une maquette exécutable à partir des spécifications DSL. Par maquette nous entendons une réalisation à échelle réduite du S.I. projeté. La réduction concerne les points suivants: le système généré est considéré comme monoutilisateur, pas de prise en charge des problèmes de fiabilité d'exploitation (création de copies de sécurité, gestion des reprises, ...) pas de recherche de performances d'exploitation, le logiciel travaille en volume réduit de données.
L'utilité de ce logiciel est de permettre aux utilisateurs de se rendre compte rapidement si les spécifications du futur S.I. correspondent bien à leurs besoins. Le comportement attendu est traduit sous la forme de plan de tests. L'exécution de la maquette générée à partir du dictionnaire engendre le comportement spécifié.
La comparaison des deux décèlera les spécifications à revoir.

d. L'environnement en place à la CGER

Il serait intéressant à ce point de l'étude de faire quelques comparaisons entre l'environnement logiciel DSL et les outils utilisés à la CGER.
Le dictionnaire DATAMANAGER du point de vue de ses fonctionnalités correspond à peu près à l'environnement DSL/SPEC:

- langage de définition des données composé de clauses descriptives des différents types de membres et des clauses relationnelles permettant d'introduire des relations entre membres.
- langage de manipulation des données constitué de commandes de manipulation de données permettant d'insérer, de modifier, de supprimer des membres dans la base de données et de commandes d'interrogation de données permettant de connaître les différentes relations entre les membres.

Le principal intérêt du dictionnaire est de retrouver aisément les informations à traiter par un processus quelconque.
Comme nous l'avons signalé plus haut, la différence principale réside dans

les possibilités graphiques utilisables pour l'acquisition, la modification et la restitution des données.
Un outil CASE (Computer Aided Software Engineering) possédant les mêmes possibilités graphiques existe à la CGER: EXCELERATOR. Cet outil possède son propre dictionnaire de données et permet de construire six types de schémas:

- diagramme de flux de données
- les "structure charts"
- les "structure diagrams"
- les modèles logiques de données
- les schémas E/A
- les "presentation graphs".

Les données introduites pourront être contrôlées à partir des schémas élaborés. Des contrôles de complétude et de cohérence peuvent être effectués. Les concepteurs pourront également directement utiliser les données du dictionnaire pour l'élaboration des divers documents.
Les groupes outils et méthode sont en train d'élaborer une passerelle entre le dictionnaire de données et l'outil EXCELERATOR car les données introduites dans EXCELERATOR n'étaient pas récupérables par DATAMANAGER. Ce manque d'interface découragait l'emploi d'EXCELERATOR car les informations devaient obligatoirement être introduites dans son dictionnaire de données et n'étaient pas récupérables. Lorsque le lien XL-DMR sera réalisé, XL (qui n'était jusqu'ici employé qu'à titre expérimental) sera propagé.

PALOMA est un outil qui a été développé à la CGER et qui fournit toutes les informations nécessaires à la construction du modèle dynamique des données. Il valide également les modèles conceptuel des données et des traitements par une confrontation mutuelle. Enfin, PALOMA estime la charge représentée par les tâches du système. Il est du point de vue de ses fonctionnalités fort proche du DSL/SIM bien que plus axé sur le modèle conceptuel des données que sur la dynamique des traitements.
" PALOMA estime la charge supportée par chaque arc reliant les divers objets dans le modèle conceptuel des données en fonction du nombre de tâches qui le parcourent, du nombre maximum d'exécutions de ces tâches et du mode d'accès à l'extrémité de l'arc: lecture, modification, insertion et suppression . 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'occurences d'objets mises en oeuvre et de la manière dont on accède à ces objets". (M.A.I., p.3).
les informations que fournira 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

- une liste condensée résumant la charge des tâches
- une liste détaillée explicitant le parcours de chaque tâche
- un listing d'erreurs

PALOMA utilise les informations contenues dans le dictionnaire de données. Il sert à valider le schéma E/A. Il peut par exemple signaler les associations qui, selon lui, manquent lorsque par exemple, lors de l'estimation du trafic, il constate un accès fréquent de l'entité B à l'entité A alors que les deux entités ne sont pas reliées par une association. Le schéma devra dans ce cas parfois être complété. L'estimation des trafics et des points d'entrée dans le modèle va également aider considérablement le DBA à élaborer le modèle logique.
Les traitements pourront être réorganisés si les estimations faites par PALOMA s'avèrent trop lentes ou inexécutables pour diverses raisons.

Sans pousser plus loin la description, nous pouvons constater que le département informatique dispose d'outils puissants (DMR, PALOMA) , conviviaux (EXCELERATOR) pour les différentes étapes du développement d'un logiciel, mais que le problème majeur est celui de l'interface imparfaite entre ces outils. Comme tous les développements doivent utiliser le dictionnaire de données DMR (entre autre pour passer par l'étape PALOMA et pour le passage en production), et que celui-ci est puissant mais peu convivial (Il manque par exemple à DATAMANAGER des utilitaires d'acquisition, de modification, d'extraction et de restitution de données graphiques tels que ceux de DSL/SPEC), les développeurs sont peu motivés à réintroduire les données dans d'autres outils.
DMR étant le coeur de pratiquement toutes les applications sur mainframe tournant à la CGER, il est impossible de faire subir une conversion de toutes ses informations. Il faudrait au contraire pouvoir venir faire greffer sur le dictionnaire des outils compatibles pouvant exploiter et modifier les données du dictionnaire comme cela est en train de se faire pour EXCELERATOR (éditeur graphique, outils de prototypage, générateurs d'applications, ...).
Le développement interne de tels outils représenterait une charge de travail trop importante. La CGER devra donc se tourner vers le marché et se procurer au fur et à mesure des outils pouvant communiquer entre eux et guidant les développeurs à travers les différentes phases des développements. Ceci correspond du reste à la philosophie AD/CYCLE d'IBM.
En attendant ce support de développement, le département informatique est en train de réaliser une interface entre son dictionnaire de données et l'outil graphique EXCELERATOR. Il se dirige donc vers des outils de développement plus proches de ceux du type DSL.

3.4. Suggestions organisationnelles

Sur base des interviews, nous ferons pour terminer cette partie quelques suggestions d'amélioration de circulation de l'information entre les différents groupes.

Des réunions régulières entre les divers groupes de support et les groupes de développement existent (les steering commitees development), elles permettent un échange d'informations bilatéral: les groupes de support y présentent les nouveaux produits, y exposent les divers problèmes revenant fréquemment, y fournissent des conseils quant à l'utilisation de certaines méthodes ou techniques. Les groupes de développement d'autre part ont l'occasion d'y poser des questions à propos des méthodes ou produits dont s'occupent les groupes de support. Ces réunions, contrairement à la proposition faite plus haut, (participation d'un représentant de chaque groupe support à au moins une réunion de chaque groupe de développement) exigent moins de charge de travail de la part des groupes de support, mais elles répondent également moins bien au besoin de chaque développeur en particulier car seuls y participent le premier conseiller, les deux informaticiens D, les chefs de service et les informaticiens outils et méthodes. Nous avons en effet constaté que plusieurs analystes et plusieurs programmeurs (qui ne participent pas à ces comités) réclament plus de contacts avec les groupes de support. Une plus grande participation des membres des groupes de support à chaque projet (réunions périodiques avec tous les développeurs participant à un projet) répondrait aux demandes exprimées: les réunions pourraient alors être formalisées et les contacts entre groupes se feraient moins "sur le terrain".

Les groupes méthode et outils font régulièrement paraître des brochures reprenant l'objet de leurs travaux (method news et tools info) et les envoient à tous les chefs de service. Les résultats des interviews montrent cependant que des analystes et des programmeurs ignorent complètement l'existence de ces brochures et réclament leur création. S'il est impossible d'envoyer ces parutions à tous les développeurs, il faudrait au moins que ceux-ci soient prévenu de leur existence et que les brochures soient mise à la disposition de tous en des endroits précis et connus de façon à ce que leur consultation puisse se faire aisément.

Une troisième suggestion concerne la communication inter-groupes de développement. Comme le département informatique ne cesse de croître et que les communications entre chaque développeur sont devenues impossibles, il

est fort probable que des problèmes divers surgissent dans un groupe de travail, que celui-ci trouve les solutions et que plus tard les mêmes problèmes se posent dans un autre groupe de travail. La communication inter-groupes n'étant pas toujours formalisée (en tous cas pas entre les développeurs non informaticiens) les informations et les solutions ne parviennent pas toujours aux groupes intéressés. Même si les forums, les réunions de discussion autour d'un thème répondent en partie à ce problème, des réunions mensuelles horizontales (inter-groupes de développement) comprenant non seulement les coordinateur de projet (comme cela existe aujourd'hui) mais aussi des représentants des informaticiens, des analystes et des programmeurs de chaque groupe de développement, malgré la charge de travail supplémentaire, semble un investissement rentable.
Les différents points de vue, les problèmes rencontrés et les solutions qui ont été appliquées seront ainsi communiqués à tous les intéressés.

Nous voyons ici que la méthode de développement SDM n'est qu'un élément d'une panoplie de moyens qui permettent le fonctionnement harmonieux d'un département dont l'environnement devient de plus en plus complexe et de plus en plus instable.
Un autre élément, qui n'a jusqu'ici pas été incorporé dans SDM, est la mise en place de structures organisationnelles favorisant la circulation de l'information. Celle-ci exige en effet des structures particulières lorsqu'une organisation devient importante car, vu le nombre de ses membres, les communications entre chacun d'eux deviennent impossibles. Des structures devront être mises en place pour formaliser et pour canaliser les divers flux d'informations. Elles pourraient ainsi faciliter la communication de l'information en spécifiant qui doit communiquer quoi à qui.

Vers une structure plus bureaucratique?

A la fin du chapitre 1, nous avons remarqué que le département informatique de la CGER (ou, plus précisément, son département développements) se rapproche fort de la structure organisationnelle adhocratique décrite par H. MINTZBERG (1982). Nous avons également cité les avantages et les inconvénient de cette structure.
Sur base de tout ce qui précède, pouvons nous affirmer que SDM va rapprocher le département informatique vers un autre type de structure organisationnelle ?
SDM va influencer la formalisation des comportements en proposant les différentes étapes par lesquelles un développement doit idéalement passer, elle va également réduire le nombre des communications informelles car elle est, comme nous l'avons déjà souligné, un coordinateur de passage d'information.
Toutefois, son application laisse assez de souplesse et d'autonomie aux développeurs pour faire face aux demandes de développement qui leur parviennent, elle propose une méthode qui peut être adaptée en fonction des différents projets .
Nous ne pouvons donc pas conclure que SDM va réellement "bureaucratiser" le département.

CONCLUSIONS

Comme la plupart des activités humaines, la conception et le développement de systèmes d'information ne semblent pas échapper à la standardisation et à l'automatisation.
Le passage d'un stade artisanal à un stade industriel semble aujourd'hui bien entamé et est devenu inévitable.
Le stade artisanal se caractérisait généralement par :

- des développements "peu efficients"
- une grande dépendance vis-à-vis de l'expérience et du savoir faire des développeurs
- des développements peu standardisés et peu documentés
- des maintenances et des ajouts très difficiles et très longs à réaliser

Le stade industriel possède généralement les caractéristiques contraires.
Le passage de l'un à l'autre est tout d'abord possible par l'application d'une méthode de développement théorique et par des outils de développement de logiciels puis par un outil de support intégré (outil qui n'existe pas encore en ce début des années 1990).
Le département informatique de la CGER est en train de vivre ce passage grâce à la méthode SDM.
L'implantation de celle-ci s'est faite de manière non autoritaire et progressive. Il en résulte que tous les développeurs ne l'adoptent pas encore.
Les développeurs favorables à SDM ont estimé que les facteurs encourageant sont emploi sont:

- la standardisation des développements: un des effets les plus importants de SDM est le passage de logiciels élaborés selon la méthode de chaque développeur vers des logiciels élaborés selon une méthode commune
- souplesse d'utilisation. La méthode SDM a été adaptée à l'environnement informatique de la CGER. Il en résulte entre autre une "checklist" reprenant toutes les étapes d'un développement précisant pour chacune d'elles les responsables à contacter et les outils et méthodes disponibles. Cet aspect liste exhaustive permet une grande souplesse d'utilisation: en fonction des projets, les développeurs pourront passer les étapes qui ne sont pas applicables ou qui n'ont pas d'intérêt.
- guidance dans le travail. SDM propose une liste d'étapes à suivre pour développer un logiciel. Les développeurs ont apprécié cette guidance, cette aide qui leur est accordée pour élaborer leur travail

- prise en compte de tous les éléments devant intervenir lors de l'élaboration d'un logiciel: aspect de sécurité, aspect confidentialité de l'information, prise en compte des capacités machine et mémoire,...

Les développeurs ont également mis en avant les facteurs décourageant l'emploi de SDM:

- si SDM constitue une bonne guidance et laisse une grande part de créativité aux développeurs, elle ne détaille pas beaucoup les étapes les moins standardisables d'un développement
- SDM préconise beaucoup de travail administratif: analyses, documentation
- SDM n'est pas soutenue par un outil intégré
- SDM étant une méthode générale applicable à tout type de développement, elle n'est adaptée à aucun d'eux

Les inconvénients cités peuvent être considérés comme des "maladies de jeunesse" de la méthode.
Les groupes méthodes et outils du département informatique vont en effet, dans un futur proche, détailler la description de certaines étapes SDM et élaborer une interface entre certains outils de développement de façon à accélérer et à faciliter la conception et l'élaboration des logiciels et de leur documentation.

La perception et l'application de SDM par les développeurs sont influencées par divers éléments de l'environnement de développement. Reprenons ci-dessous les principaux:

- des tâches complexes et variées
- le besoin d'une standardisation des développements
- le besoin d'une bonne circulation de l'information
- le besoin de logiciels d'une guidance lors d'un développement
- le besoin d'une bonne interface utilisateurs/informatique
- la situation personnelle des développeurs

L'application d'une méthode de développement de logiciels engendre des produits de qualité mieux conçus, mieux analysés et mieux documentés.

Le stade final reste toutefois encore à atteindre à la CGER: celui où la méthode SDM sera inconsciemment appliquée par les développeurs.
Ceux-ci travailleront alors selon les normes SDM sans le savoir, comme s'il s'agissait d'une méthode qui leur est propre et qu'ils estiment la meilleure.

POSTFACE

Dans ce qui précède, j'ai tenté de rester le plus objectif possible de manière à ne pas fausser l'enquête et les analyses par des considérations personnelles. Je tiens cependant à reprendre dans cette postface quelques réflexions que je me suis faites lors de leur élaboration.

1. Position a priori

N'ayant jamais eu l'occasion d'appliquer une méthode de développement de logiciels au moment du choix du sujet de mon mémoire, mon sentiment à leur propos est résumé restait vague. Je l'ai repris ci-dessous.
Les directions générales des entreprises dès que les budgets informatiques prennent une certaine ampleur vont exiger des responsables informatiques de limiter les coûts. Comme les coûts salariaux constituent une part importante d'un budget informatique, la productivité des développeurs va être un des aspects sur lesquels les responsables vont essayer d'intervenir.
Une méthode de développement de logiciels va dans le sens d'une réduction des coûts de développement et d'exploitation d'un logiciel et cela pour différentes raisons:

- le travail de développement sera standardisé, les étapes à suivre sont fixées ainsi que la documentation à établir. Il en résultera des programmes mieux standardisés et mieux documentés dont la maintenance sera plus facile.
- Cette standardisation des étapes va permettre d'exercer des points de contrôle sur la qualité et sur la rapidité du travail des développpeurs.
- Une plus grande standardisation des développements va également diminuer la dépendance d'une entreprise envers ses développeurs. Comme ceux-ci ne pourront plus développer "selon leur méthode", leur travail deviendra plus transparent autant pour leurs supérieurs hiérarchiques que pour leurs collègues.

D'autre part, le sentiment des développeurs vis-à-vis d'une méthode de développement sera négative:

- le développement d'un S.I. est une activité exercée par des spécialistes qui n'ont pas l'habitude que l'on vienne s'occuper de la façon dont ils travaillent.
- Le travail de développement requiert une large part de créativité car chaque logiciel à élaborer est unique, il faut toujours chercher de nou-

velles solutions pour résoudre le problème. Une méthode de développement sera pour cette raison ressentie comme une limite à la créativité, comme un encadrement rigide des activités de développement.
- Les développeurs pourraient être hostiles à la méthode simplement parce qu'elle est neuve et qu'ils ne la connaissent pas (crainte de l'inconnu).

2. Une méthode de développement de logiciels à la CGER

Après avoir pris connaissance de l'environnement informatique de la CGER, il me semble que la méthode a été choisie pour des raisons citées plus haut (réduction des coûts de développement, standardisation du travail de développement,...) , mais également pour d'autres auxquelles je n'avais pas songé au départ:
- prise en copmpte des aspects de sécurité, de confidentialité des informations,...
- une meilleure qualité des logiciels (bonne maintenabilité, fiabilité, performances,...)
- une meileure aide fournie aux développeurs.

Nous verrons plus loin dans quelle mesure ces différents objectifs ont, à mon avis, été atteints.

3. La méthode SDM

L'objet de ce mémoire n'était pas d'analyser la méthode SDM en profondeur comme cela l'a été fait dans [C. LEFEBVRE et J.-N. PHILIPPOT (1989)] ni de comparer la méthode avec d'autres. Si l'on devait classer la méthode dans une famille particulière, et en reprenant la typologie proposée dans [N. DELAHOUSSE et P. VANCUTSEM (1988)] , SDM prendrait place dans la famille socio technique: " *cette famille préconise de tenir compte simultanément du sous-système technique et du sous système social: il ne suffit pas d'élaborer un bon sous-système technique et d'y adapter le sous-système social ou inversement. Si l'on ne prend pas en compte à la fois les aspects sociaux (y compris les aspects organisationnels qui y sont très liés) et les aspects techniques du S.I. lors de son développement, certains éléments seront négligés et il sera trop tard pour essayer de les intégrer ultérieurement."* (p. 112)
Les aspects sociaux et organisationnels sont principalement pris en compte lors des étapes SDM suivantes:

1.2. rassembler les données sur la situation existante

1.4. déterminer les objectifs et les exigences du nouveau système

2.2. déterminer le cadre dans lequel le nouveau système devra fonctionner
2.8. déterminer les problèmes humains et les solutions.

Ces étapes sont établies en collaboration étroite avec l'utilisateur et en parallèle avec les problèmes techniques.

4. A quelles demandes des développeurs SDM répond-elle ?

Les plaintes les plus souvent exprimées par les développeurs concernent:

- le dialogue avec l'utilisateur
- le manque de souplesse et d'intégration des outils
- le manque de précision dans la définition des tâches (adaptation de SDM à chaque poste de travail)
- l'insuffisance des contacts avec les groupes de support.

Reprenons un à un ces différents points.

SDM préconise une étroite collaboration utilisateurs/informatique. Il m'a toutefois semblé que avec ou sans méthode, cette collaboration était fructueuse lorsque l'utilisateur savait ce qu'il voulait et était conscient des possibilités et des limites de l'informatique et que, par contre, elle était mauvaise dans le cas contraire.

Il faudrait pour résoudre ce problème que tous les représentants des utilisateurs suivent des journées de formation à divers aspects de l'informatique (méthodes, outils, techniques,...). Le groupe méthodes (qui prend également en charge l'organisation des formations) est conscient du problème, mais ne peut faute de moyens, faute de personnel et peut être aussi faute d'intérêt de la part des utilisateurs assurer des formations pour tous les représentants des utilisateurs.

Dans une organisation pour laquelle la rentabilité est un élément primordial, il est étrange de ne pas voir à côté des groupes méthodes et outils un groupe formation informatique qui pourrait s'occuper à temps plein de l'organisation des formations et, entre autres, des formations utilisateurs qui semble un investissement rentable.

Les plaintes concernant le manque d'intégration et le manque de souplesse des outils sont fondées puisque nous avons vu que des outils plus souples et plus intégrés existent. La CGER attend toutefois le repository d'IBM, sorte de dictionnaire de données sur lequel viendraient se greffer différents outils correspondant aux différentes phases d'un développement de logiciel. En attendant, le département développe lui-même des passerelles entre ses outils.

Nous avons déjà mentionné à plusieurs reprises l'importance capitale des outils en tant que support d'une méthode de développement. Le manque d'outils intégrés semble aujourd'hui la maillon faible de l'application de la méthode.

Les groupes méthodes et outils en ont pris conscience et y remédient dans la

mesure de leurs possibilités.
Les demandes concernant les définitions de tâches plus détaillées et une adaptation de SDM à chaque poste de travail suggèrent quelques réflexions.
Tout d'abord, les tentatives qui ont été entreprises dans ce sens ont engendré des réactions fort hostiles de la part des développeurs. Ils craignaient probablement les conséquences négatives qui pourraient résulter d'une définition précise de leur poste de travail (contrôles accrus, liberté moindre,...).
D'autre part, les développeurs en réclamant une plus grande définition des tâches ou une adaptation de SDM à leur poste de travail réclament en fait une méthode de développement "toboggan". Si une telle méthode voyait le jour, l'utilité des développeurs serait pratiquement réduite à néant.
Pour ce point également, la meilleure solution semble être les outils de développement. Ceux-ci n'engendreraient pas de réactions d'hostilité à une définition précise de chaque poste de travail ou à une méthode de développement très détaillée et guideraient les développeurs en leur proposant une série de démarches à suivre.
Abordons maintenant les plaintes concernant les contacts trop peu nombreux avec les groupes de support.
Pour atteindre la situation idéale où un membre de chaque groupe de support devrait suivre chaque projet, il faudrait gonfler les effectifs de ceux-ci. Actuellement, un tel suivi est d'autant plus impossible que les groupes méthodes et outils doivent également s'occuper d'autres activités prioritaires telle que l'élaboration de passerelles entre outils de développement.
Il serait intéressant de calculer si les coûts provenant de contacts insuffisants entre développeurs et les groupes de support (utilisation non optimale d'outils et méthodes, méconnaissance d'outils de techniques,...) sont supérieurs à ceux qu' engendrerait une augmentation des effectifs.

5. Réactions des développeurs face à la méthode

Essayons de voir dans ce paragraphe les raisons pour lesquelles, à mon avis, les différents types de développeurs réagissent plus ou moins positivement à la méthode.

a. <u>les informaticiens</u>

Les informaticiens ont pour tâche la gestion des projets. En tant que chefs de projet, ils sont responsables de la qualité et des délais de dévelopemment des logiciels dont ils ont la charge. Ce sont également les développeurs le plus en contact avec les utilisateurs.
Quelles pourraient dès lors être leurs réactions face à une méthode de développement ?
Tout d'abord, comme nous l'avons déjà mentionné, la diversité des tâches qui incombent aux informaticiens est telle qu'une liste exhaustive et descriptive de

ces différentes tâches est la bienvenue.
Ensuite, les informaticiens sont la catégorie de développeurs étant responsables de la qualité des logiciels vis-à-vis de l'extérieur et plus particulièrement vis-à-vis des utilisateurs. Si un problème se pose lors des tests d'acceptation ou par la suite, ce sont les informaticiens qui sont tenus pour responsables.
Une méthode de développement, si elle est bien appliquée, garantit une plus grande qualité des logiciels: les besoins des utilisateurs ont été bien analysés, les éléments de sécurité, de confidentialité, de capacité machine ont été pris en compte, les analyses ont demandé un certain temps de réflexion, la documentation est complète (la maintenance pourra se faire plus rapidement). Ce deuxième point devrait donc également engendrer des réactions positives de la part des informaticiens.
Un troisième facteur allant dans ce sens est la position même qu'occupent les informaticiens. Vu leur grade professionnel et leurs activités, les informaticiens sont les développeurs les plus susceptibles de s'intéresser aux méthodes de développement: ils ont des contacts fréquents avec les groupes de support, ils sont responsables du travail de leurs analystes et de leurs programmeurs, ils ont également à accomplir beaucoup de tâches administratives.
Par contre, en tant que chefs de projet, les informaticiens doivent veiller à ce que les délais et les coûts de développement soient respectés.
Une méthode de développement rigide et monolithique pourrait donc être mal perçue. SDM en proposant une liste d'étape semble échapper à ces critiques puisque, si les délais de développement sont très courts, si le budget est très serré, il sera parfaitement possible de passer certaines étapes moins primordiales.

La réaction des informaticiens vis-à-vis de la méthode devrait donc être largement positive. Si l'on reprend les chiffres du degré d'application de la méthode, nous constatons en effet que 6 des 7 informaticiens développeurs appliquent SDM soit complètement (3) soit sa philosophie (3).

b. Les analystes
Comparativement aux informaticiens, les analystes sont de part leurs tâches moins sensibles aux aspects administratifs des projets.
Plus proches des aspects techniques, leurs préoccupations quant aux délais et aux coûts à respecter, aux rapports et à la documentation à élaborer peuvent dès lors leur paraître secondaires. Ils sont plus enclins à faire de l'informatique efficace que de l'informatique efficiente.
Une liste d'activités à accomplir peut cependant les guider dans leurs tâches.
De plus, comme la méthode peut être appliquée souplement, les analystes n'ont pas à craindre les inconvénients d'une méthode rigide: points de con-

trôle, diminution de la créativité, autonomie moindre,... .
Si nous prenons les résultats du degré d'application de SDM par les analystes, nous constatons que des 7 analystes développeurs, 1 l'applique complètement, 2 en appliquent la philosophie, 1 ne l'applique pas et 3 ne connaissent pas SDM.
Le degré d'application est donc nettement inférieur à celui des informaticiens
Les raisons que nous venons de citer peuvent nous aider à comprendre pourquoi.

c. Les programmeurs
Il s'agit de la catégorie de développeurs la plus proche des aspects purement techniques et la plus éloignée des considérations de gestion des projets.
La lourdeur des rapports et la documentation volumineuse à établir leur paraîtra de ce fait un inconvénient majeur de la méthode. Les programmeurs verront également plus facilement une méthode de dévelopement (ne fut-ce que par méconnaissance de celle-ci) comme un danger menaçant leur autonomie.
Etant les développeurs dont les préoccupations sont les plus éloignées des méthodes de développement de logiciels et les plus proches des aspects de programmation, la crainte, les réactions négatives ou le désintérêt vis-à-vis d'une méthode qu'ils ne connaissent pas ou mal seront les plus fortes.
Les résultats de l'enquête confirment ce qui précède: des trois programmeurs, deux ne connaissent pas SDM et un ne l'applique pas.

A côté du grade professionnel, l'ancienneté joue également un rôle important: les développeurs engagés depuis peu à la CGER subiront l'influence de trois facteurs supplémentaires:

- leur formation les aura initié aux méthodes de développement de logiciels, ils seront donc plus conscients de leur utilité et de leur nécessité
- le nouvel environnement dans lequel ils se trouvent est complexe et dynamique. Une guidance méthodologique des activités qu'ils ont à accomplir sera utile sinon nécessaire
- une personne travaillant depuis peu dans une organisation sera beaucoup plus encline à accepter et à adopter une méthode de travail qui lui est proposée car elle n'a pas encore acquis d'expérience et de méthode de travail dans le nouvel environnement

Ces trois facteurs engendrent des réactions favorables à SDM.

- les développeurs qui par contre sont engagés à la CGER depuis plusieurs années subiront l'influence des facteurs contraires:

- leur formation, surtout s'ils sont plus âgés, ne les a pas initié aux méthodes de développement

- ils ont acquis des habitudes de travail, ils seront donc plus réticents à en changer et à adopter une méthode de développement
- ils ont une meilleure connaissance de l'environnement informatique et des produits financiers et ressentiront par conséquent moins la nécessité d'être guidé.

Les considérations qui précèdent sont toutefois limitées:
quel que soit le grade professionnel ou l'ancienneté des développeurs, leur caractère va intervenir de façon importante.
Certaines personnes n'aiment par exemple pas être guidées de près. Certaines personnes ne sont jamais satisfaites de leurs conditions de travail et se plaignent à tort et à travers tandis que d'autres ne se plaignent jamais alors qu'elles ont de bonnes raisons pour le faire.
Certaines personnes critiquent pour le plaisir de critiquer, sans proposer de suggestions constructives lorsqu'on cherche à les aider (cela explique en partie le taux de réponse très faible à certaines questions de l'enquête).
La prise en compte de ces éléments exige une bonne connaissance des différentes personnes, je n'ai pas pour cette raison pu en tenir compte dans mon enquête.

6. Le rôle des groupes méthodes et outils

Le rôle principal des ces groupes est de proposer aux divers groupes de développement des méthodes, des techniques et des outils permettant des développements de qualité, rapides et au coût le plus faible possible.
Si les préoccupations méthodologiques semblent très poussées comparativement aux autres entreprises (voir à ce propos l'enquête menée dans L'état de l'informatisation des entreprises, 1989, pp69-71), les moyens et l'autorité des groupes méthodes et outils ne m'ont pas semblés à la hauteur de l'ambition (une méthode de développement adoptée par tout le monde et soutenue par des outils les plus intégrés et les plus souples possibles). J'ai pu constater d'après les plaintes exprimées que beaucoup de choses restaient à faire, que les groupes méthodes et outils en sont conscients, mais, faute de moyens, des priorités très strictes doivent être établies.
Un aspect plutôt négligé de part ce fait est le "marketing" de la méthode. Les limites budgétaires des groupes de support ne leur permettent pas de "vendre" SDM , même s'ils organisent des cours et des forums, l'enquête nous a montré que SDM est loin d'être connue par tous les développeurs.
Nous pouvons nous demander à ce sujet s'il ne serait pas rentable, même si à première vue cela semble difficile à mettre sur pied, d'arrêter pendant quelques semaines toute activité de développement afin de permettre à

chacun d'acquérir une connaissance approfondie de la méthode, de faire part de ses questionsl ainsi que de ses suggestions d'amélioration.
Cela pourrait par exemple se faire sous forme de cercles de qualité où chacun réfléchirait aux divers problèmes méthodologiques et pourrait apporter sa contribution.
Le terrain serait "dégivré" après quelques semaines, tout le monde y aurait acquis un langage commun et de solides fondations méthodologiques sur lesquelles il serait plus aisé de construire une méthode appliquée par tous.

7. Des objectifs atteints ?
Dans quellesmesure les objectifs deSDM ont-ils été atteints ?
La standardisation des logiciels est plus grande lorsque la méthode est appliquée: les étapes, les analyses, l documentation à établir et parfois même la manière de procéder sont décrites par SDM.
Toutefois, deux problèmes subsistent:

- la méthode n'est pas appliquée par tout le monde
- le temps de développement ne permet pas toujours d'appliquer toutes les étapes intéressantes pour le projet.

Une meilleure qualité de logiciels semble acquise lorsque SDM est appliquée:

- bonne maintenabilité grâce à la documentation, à des programmes bien structurés
- bonne fiabilité grâce à des besoins utilisateur bien analysés et à des tests bien menés
- bonnes performances grâce à un effort d'optimisation

La rapidité de développement et le coût de développement lorsqu'on applique la méthode sont pour l'instant toujours freinés par l'absence d'outils intégrés et souples.

Pour terminer, soulignons qu'une méthode de développement de logiciels ne fera pas d'un mauvais un bon développeur, son rôle se limitera à fournir à ceux qui l'emploient un cadre, une guidance à travers les différentes étapes.

A la suite de l'enquête, ma perception des méthodes de développement a été quelque peu modifiée.
Elles me semblent fort utiles pour la qualité des logiciels et la bonne communication entre développeurs d'une part, entre développeurs et utilisateurs d'autre part.
Ces méthodes ne semblent pas mettre en péril le métier de développeur car, même si un développement peut être canalisé, il nécessitera toujours une grande part de créativité et de connaissances non automatisables.
Ces méthodes enfin peuvent aider les développeurs à travers la multitude de

tâches qu'ils ont à effectuer.

BIBLIOGRAPHIE.

BODART, F. , et PIGNEUR, Y. , **Conception assistée des systèmes d'information**, (Masson, 1989).

BORUM, F. , **Beyond Taylorism: The IT-Specialists and the deskilling hypothesis**, (september 1987).

CHANDLER, M. , K. , and SAYLES, L. , R. , **Managing large Systems** (Harper and raw, 1971).

de BRUYNE, P. , **Théorie des Organisations**, (CIACO, 1987).

DECUYPER, B. , GOSSELIN, Fr. , LOBET-MARIS, Cl. , VERDURE, J.-M. , **L'état de l'informatisation des entreprises**, journal de réflexion sur l'informatique, (mai 1989).

DELAHOUSSE, N. , VANCUTSEM, P. , **Méthodes de conception et de développement de système d'information**, (mémoire, Institut d'informatique, Namur, 1988).

FRIEDMAN, A., **Understanding the employment position of computer programmers : a managerial strategies approach**, CHIPS (september 1987).

Groupe méthodes du DSIM, **De SDM methode in de ASLK**, (document DSIM, 1989).

HAINAUT, J.-L. , **Conception assistée des applications informatiques**, (Masson, 1988).

IEEE Workshop on Software Productivity Issues, Fifth International IEEE Symposium on Software Engineering (march 1981).

JONES, C. , **La productivité en génie logiciel**, (les éditions de l'organisation, 1989).

KHANDWALLA, P. , N. , **Organisational design for change**, Conceptual Reading (1976).

KNIGHT, K., **Matrix organisation: a Review**, The journal of Management Studies (1976)

LAWRENCE, P. , R. , **Organizations and environment**, (Harvard University, Boston, 1967).

LEFEBRE, C. , PHILIPPOT, J.-N. , **Contribution à l'introduction d'une méthodologie de développement dans une grande entreprise**, (mémoire, Institut d'informatique, Namur, 1989).

M.A.I. , **Répartition des tâches entre analystes et programmeurs dans un environnement J.S.P.**, (document interne CGER)

Mc GREGOR, D., **The human side of Enterprise**, (Mac Graw-Hill, NY, 1960)

MINTZBERG, H., **Structure et dynamique des organisations**, (Les Editions de l'organisation, 1982)

SAYLES, L.,R., **Matrix Organisation: the Structure with a Future**, Organisational Dynamics (autumn 1976)

SIMULA, P., **Les Emplois de l'Informatique**, Les professions de l'informatique, dossier professionnel, CEREQ, La Documentation Française (juin 1986, pp. 81-215)

CGER, les vertus privées d'une banque publique, TRENDS-TENDANCES (14 avril 1987).

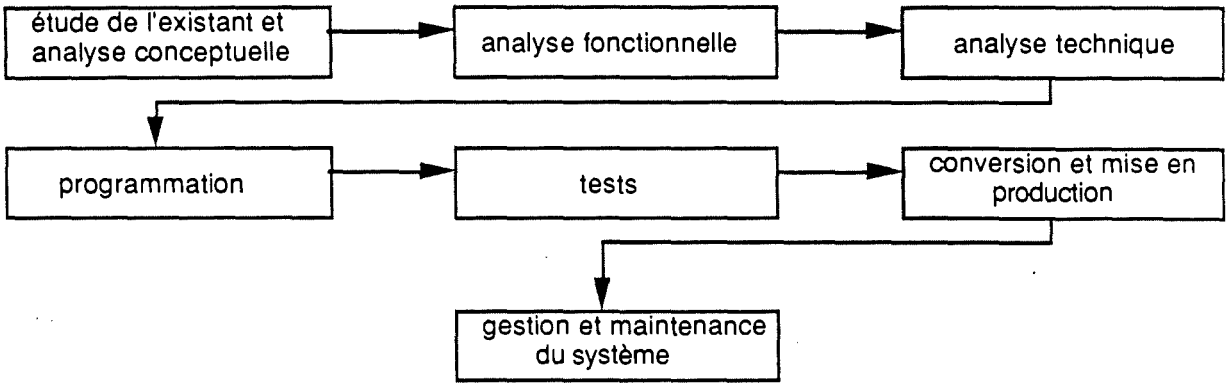
WOODWARD, J. , **Industrial Organisation: The Organisation and Practice** (Oxford University Press, 1965).

ANNEXE 1

QUESTIONNAIRE UTILISATEUR.

Situation personnelle.

1) Quel est votre grade professionnel ?
2) Quelle est votre formation de base ?
3) Depuis quand travaillez-vous à la CGER et depuis quand occupez-vous votre poste actuel ?
4) Pourriez-vous décrire brièvement en quoi consiste votre travail ?
5) La méthode SDM reprend les sept phases de développement suivantes:



D'après la description de ces phases ,quelles sont les activités auxquelles vous participez ?

6) Pour chacune d'elles,pourriez-vous expliquer:
   - ce que vous y faites et avec quels outils
   - quels sont les points de contrôle par lesquels vous passez
   - s'il y en a ,les points qui à votre avis pourraient être améliorés concernent:
     - les outils utilisés
     - les consignes quant à votre travail
     - les méthodes de développement de logiciel
     - les contrôles ou points de passage
     - le manque de temps ou,au contraire,l'importance des temps morts
     - le travail en équipe
     - autres.

   Concernant ces points,pourriez-vous formuler des propositions d'amélioration ?
7) Par ordre décroissant d'importance, pourriez-vous citer trois changements techniques et trois changements organisationnels qui ont affectés votre travail au cours de ces dernières années ?
8) Avez-vous suivi des formations en 1989 ?
9) Si oui,quels ont été les cours les plus utiles pour la réalisation de votre tâche ?

Organisation et répartition du travail au sein des groupes d'étude.

10)Qui répartit et qui coordonne le travail à l'intérieur des groupes d'étude auxquels vous participez ?
11)Comment cette répartition est-elle effectuée ? (en fonction des spécialités de chacun ,en fonction des disponibilités,...).
12)Y a-t-il des propositions que vous aimeriez formuler concernant la circulation des informations et des consignes de travail ? Pourriez-vous justifier ces propositions en citant quelques exemples ?
13)Pensez-vous que des améliorations pourraient être apportées à la répartition du travail au sein des groupes d'étude ou à l'organisation de votre propre tâche ?
14)Y a-t-il des contacts entre groupes d'étude? Ces contacts sont-ils informels ou ont-ils été formalisés ?
15)L'influence que vous pouvez exercer sur :

- la définition de votre travail
- l'organisation de votre travail
- le temps de développement d'une application
- les outils que vous utilisez
- les points de passage
- les méthodes de développement de logiciel.

est-elle limitée, importante, très importante ?
16)Quels sont les deux domaines pour lesquels ,selon vous, une aide ou une information supplémentaire seraient les mieux venues:

- information sur les produits bancaires et d'assurance
- outils de développement
- définition des tâches
- consignes de développement
- autres

17)Pour chacun de ces deux domaines,par qui et comment cette aide serait-elle le plus efficacement fournie ?

La méthode SDM.

18)Selon vous,comment la collaboration avec les services de support (préparation, production, bureautique, méthodes, outils, etc,...) pourrait-elle être améliorée ?
19)D'après vous,quels éléments de méthode,d'outil,de guidance devraient être ajoutés, réorganisés ou allégés dans les étapes de SDM afin d'accélérer le développement ?
20)En particulier,suggérez-vous des adaptations dans les cas suivants:

- achat de progiciels
- sous-traitance
- développement sur mini-ordinateurs
- etc,...

21)Selon vous ,quels sont les principaux points forts de SDM ?
22)Pensez-vous que SDM comporte des points faibles ? Pourriez-vous justifier votre réponse ?
23)De quelles informations supplémentaires aimeriez-vous disposer à propos de SDM ?
24)Pensez-vous que SDM va modifier votre façon de travailler ? Si oui,précisez en quoi ; sinon, pourquoi ?

25)Lorsque vous appliquez SDM,quels sont à votre avis les domaines où ses effets positifs peuvent se faire le plus ressentir ? (définition des tâches,facilité de développement, rapidité de développement,qualité du logiciel,...).

ANNEXE 2

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRO-DUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CON-CERNED PER-SONS	COURSE	ORG.	REP.	S.D.M. FAZEN
		PERSONS	TEL.	GROUP														
0. ETUDE PRELIMINAIRE		C. GRAAS	7037	DSI-M					-				Develop. Request	Project Leader				0. VOORSTUDIE
		Documentalist											SMI					
		B. BREMS	6292	PREP.									Applic. Code Assign. Form					
		P. TATON	7323	DSI-I									Accoun. Number					
		J. MATTHIJS	7247	AUDIT														
		J-F DUCHENNE	7342	Costs-centre														
		J-P. QUOILIN	7156	ORG.														
1. ETUDE DE L'EXISTANT ET ANALYSE CONCEPTUELLE							^							^				1. STUDIE VAN DE BESTAANDE TOESTAND EN CONCEPTUELE ANALYSE
1.1 Définir le problème et le champ de l'étude																		1.1 Het probleem en de omvang van de studie bepalen
1.2 Rassembler les données sur la situation existante						Interv. Tech. D.F.D. H.I.P.O. T.U.D. Dec. Tables H.S.A.T.		X						SPEC-AN MCM				1.2 De gegevens over de bestaande toestand verzamelen
1.3 Analyser et évaluer les données						Interv. Tech. D.F.D. H.I.P.O. T.U.D. Dec. Tables H.S.A.T.		X						SPEC-AN MCM				1.3 Die gegevens analyseren en evalueren
1.4 Déterminer les objectifs et exigences du nouveau système							SY-	X										1.4 De doelstellingen en de vereisten van het nieuwe systeem bepalen
1.5 Arrêter les points qui restent à résoudre et les hypothèses de base									X	X						X		1.5 De nog op te lossen punten en de gemaakte veronderstellingen vastleggen
1.6 Faire un schéma du système		C. GRAAS	7037	DSI-M		CDM/CPM H.I.P.O. Dec. Tables	SY-AC-TK-OB-AG-EL-								MCM			1.6 Een systeemschets maken
1.7 Déterminer les outils et les solutions possibles		R. MATHAR	6267	RSI/ IOTA					X	X						X		1.7 De instrumenten en mogelijke oplossingen bepalen
		D. ABRASSART	6128	PTS														
1.8 Evaluer les solutions et opérer une sélection		J. MATTHIJS	7247	AUDIT	(A.S.F.)											(X)		1.8 De oplossingen evalueren en een selectie doorvoeren
1.9 Déterminer les problèmes de conversion et de mise en production; déterminer les critères d'acceptation		R. MATHAR	6267	RSI/ IOTA					X	X	X	X				X		1.9 De problemen met betrekking tot de conversie en de invoering vaststellen; de acceptatiecriteria vastleggen
	Operability Requirements	B. BREMS	6292	PREP.														
		J-P REMORY	8386	SYS-TEMS														
		J. VAN DE WATER	8715	PROD.														
1.10 Elaborer [illegible] planning global et une vue ensemble des coûts et bénéfices					Super-Project													1.10 Een globale planning opstellen en een analyse maken van de kosten/baten
																X		1.11 Een rapport samenstellen

A.S.F.

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRODUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CONCERNED PERSONS	COURSE	ORG.	REP.	S.D.M. FAZEN
		PERSONS	TEL.	GROUP														
2. ANALYSE FONCTIONNELLE																		2. FUNCTIONELE ANALYSE
2.1 Spécifier les exigences du système, y compris futures		R. MATHAR	6267	RSI/ IOTA			SY-AC-TK-OB-AG-EL-	X	X	X								2.1 De systeemeisen specifieren, inclusief de toekomstige
2.2 Déterminer le cadre dans lequel le nouveau système devra fonctionner		L. VAN DAMME	7337	RSI/ IOTA				X	X	X						X		2.2 Het kader vastleggen waarbinnen het nieuwe systeem moet functioneren
		J-P REMORY	8386	SYS-TEMS														
		J. VAN DE WATER	8715	PROD.														
		B. HANZEN	8978	PC														
		W. DE JONGHE	8555	PC														
		D. ABRASSART	6128	PTS														
2.3 Diviser le système en sous-systèmes et les décrire						D.F.D. H.I.P.O. H.S.A.T.	SY-AC-TK-	X										2.3 Het systeem in subsystemen opdelen en deze beschrijven
2.4 Définir l'input et l'output par sous-système et les interfaces					Proto-typing (SDF II)	H.I.P.O.	OB-											2.4 Per subsysteem, de input en de output bepalen, alsook de interfaces
2.5 /6 Elaborer les diagrammes et les descriptions des traitements						D.F.D. H.I.P.O. Dec. Tables	TK-AG-EL-											2.5 /6 De processchema's en procesbeschrijvingen maken
2.7 Spécifier les exigences en sécurité et confidentialité		EDP-Security (D. HARDY)	8503	SYS-TEMS				X								X		2.7 De eisen op het gebied van de beveiliging en de vertrouwelijkheid specifieren
2.8 Déterminer les problèmes humains et les solutions																X		2.8 De menselijke problemen vaststellen en oplossen
2.9 Concevoir la structure logique des données		C. GRAAS	7037	DSI-M	PALOMA	Normalisation	OB-AC-EL-								MCM			2.9 De logische gegevens-structuur ontwerpen
	Systemchoice: IMS, DB2, ord. files,	DBA's	7348 8860	IMS/DB2											IMS			
	Turbo Image (HP), AS/400	C. DE WIT	7055	MINIS														
2.10 /12 Spécifier les facilités requises en communication de données, en hardware et en software	Production Standards Naming Conventions	B. BREMS	6292	PREP				X								X		2.10 /12 De vereiste faciliteiten met betrekking tot data-communicatie, apparatuur en programmatuur specifieren
		D. ABRASSART	6128	PTS														
	Design Rev. Walk Through	R. MATHAR	6267	AKA/BA														
		C. DE WIT	7055	MINIS														
		J-P MELEBECK	6903	PROD.	PELIKAN (File Tr.)													
2.13 Elaborer un plan pour la poursuite du développement et la mise en production	START DATUM	B. BREMS	6292	PREP.	Super-Project											X		2.13 Een plan opstellen voor de verdere ontwikkeling en de invoering
		C. DESIRON	8874	TOOLS	TIMS/ HELP													
	HP	C. DE WIT	7055	MINIS														
		M. COMHEYNS	6220	Manag.		A.S.F.												2.14 Een rapport samenstellen

EXCEL
EUDOS
USER INFORMATICIAN

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRODUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CONCERNED PERSONS	COURSE	ORG.	REP.	S.D.M. FAZEN
		PERSONS	TEL.	GROUP														
3. ANALYSE TECHNIQUE							^							^				3. TECHNISCHE ANALYSE
3.1 Conception des procédures manuelles																		3.1 Ontwerpen van de manuele procedures
3.2 Conception des formulaires et et de tous les input et output de l'ordinateur		M-C PREVOST	7758	IMS	SDF II		RP-SC-											3.2 Ontwerpen van de formulieren en van alle computer in- en uitvoer
		M. DRUEZ	7759	TOOLS	D.I.R.C.													
3.3 Conception de la structure de stockage	IMS/DB2	DBA's	7348 8860	IMS			FI-DB-RC-SG-GR-FL-	X					IMS					3.3 Ontwerpen van de opslagstructuur
		F. DE BRABANTER	7751	DB2										E				
	Ordinary Files	P. DECLERCQ	8875	TOOLS			S							S				
	Turbo Image (HP)	C. DE WIT	7055	MINIS			O							I				
3.4 Conception des mesures de sécurité	Keys	EDP-Security (D. HARDY)	8503	SYS-TEMS	R.A.C.F.		E	X						A				3.4 Ontwerpen van de veiligheidsmaatregelen
	Routines (HP)	C. DE WIT	7055	MINIS			T							N				
3.5 /6 Elaboration des descriptions de programmes et schémas	J.S.P.	M-C PREVOST	7758	IMS	SDF II	Screen Design Walk Through H.I.P.O. Pseudo-code	AP-PG-RO-PM-ME-ER-	X						A	J.S.P. SPEC-AN			3.5 /6 Programmaspecifikatie en schema's maken
		T. COURARD	8857	TOOLS	SPITAB	JSP Diagrams Dec. Tables	<PRODOS>											
3.7 Description des programmes standards à utiliser	Routines (HP)	C. DE WIT	7055	MINIS		Routines (HP)	RO-			X								3.7 Beschrijven van de te gebruiken standaard-programmas
		P. DECLERCQ	8875	TOOLS														
3.8 Elaboration d'un plan détaillé de programmation et de test					Super-Project						X					X		3.8 Opstellen van een gedetailleerd programmeer- en testplan
3.9 Rédaction du rapport						A.S.F.	v							v			X	3.9 Een rapport samenstellen

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRODUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CONCERNED PERSONS	COURSE	ORG.	REP.	S.D.M. FAZEN
		PERSONS	TEL.	GROUP														
4. PROGRAMMATION							↑							↓				4. PROGRAMMEREN
4.1 Description des tâches							TK-							ANALYST				4.1 Taakbeschrijvingen maken
4.2 Définir les exigences en personnel et environnement de travail																X		4.2 Bepalen van de vereisten qua personeel en werkomgevingen
4.3 Détailler les descriptions de programmes						Pseudo-code Dec. Tables JSP Diagrams Walk Through	PG- PCB- PM- ME-								JSP SPEC-AN			4.3 Detailleren van de programmaspecificaties
4.4 Codifier les programmes		M. DRUEZ	7759	TOOLS	ADF								X		JSP			4.4 Coderen van de programma's
	Special Software	R. MATHAR	6267	RSI/ IOTA														
		P. DECLERCQ	8875	TOOLS	COBOL II Easytrieve Plus										COBOL II Easytrieve Plus			
		B. BREMS	6292	PREP.	BETA91													
		K. TESSEUR	7094	IMS	Trans. Proc.		PRODOS							PROGRAMMER				
		M-C PREVOST	7758	IMS	Trans. Proc. (PTS)													
					ELIPS													
		C. DE WIT	7055	MINIS	POWER-HOUSE										POWER-HOUSE			
		W. DE JONGHE	8555	PC	Super DB SuperCalc5 Lotus 1-2-3										PC Products			
	Interface IMS-PTS (I9000)	D. ABRASSART	6128	PTS														
4.5 Compilation et correction					COBOL II JCL TSO													4.5 Compilatie en verbetering
4.6 Constituer les données de test						Dec. Tables					X				SPEC-AN			4.6 Samenstellen van programmatestgegevens
4.7 Tester les programmes		L. VERVAET	9118	TOOLS	Cobol Int. Debug.													4.7 Testen van de programma's
		K. TESSEUR	7094	IMS	B.T.S.													
		R. DEPREZ	8859	IMS	DLI Online													
4.8 Compléter la documentation des programmes		C. DESIRON	8874	TOOLS	TIMS/ HELP		PG- RO-											4.8 Programmadocumentatie vervolledigen

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRODUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CONCERNED PERSONS	COURSE	ORG.	REP.	S.D.M. FASEN
		PERSONS	TEL.	GROUP														
5. TESTS														^				5. TESTEN
5.1 Elaboration d'un plan de test détaillé					Super-Project			X			X							5.1 Uitwerking van een gedetailleerd testplan
5.2 Installation du matériel, du software et préparation de l'environnement de travail		P. HERMAN	6452	Techn. Inst.					X	X						X		5.2 Installatie van de apparatuur en van de software; voorbereiding van de werkomgeving
	HP AS/400	C. DE WIT	7055	MINIS														
	PC's	W. DE JONGHE	8555	PC														
5.3 Définition des unités de traitement et de leur ordre de succession																		5.3 Bepaling van de verwerkingseenheden en van de onderlinge volgorde ervan
5.4 Test des formations, utilitaires et procédures																X		5.4 Testen van de opleiding, de hulpmiddelen en de procedures
5.5 Constitution des données de test de système et d'acceptation		L. VERVAET	9118	TOOLS	(XPERT)	Dec. Tables		X			X				SPEC-AN			5.5 Samenstelling van systeem- en acceptatietestgegevens
5.6 Test des sous-systèmes		R. MATHAR	6267	RSI/ IOTA				X			X							5.6 Testen van de subsystemen
		L. VERVAET	9118	TOOLS	(XPERT)													
5.7 Exécution du test d'acceptation	Q. A.	DBA's	7348 8860	IMS		Dec. Tables		X	X		X							5.7 Uitvoeren van de acceptatietest
	Q. A.	F. DE BRABANTER	7751	DB2														
	Q. A.	E. VAN DER SYPE	6684	SYS-TEMS	T.P.N.S.													
	Special Software	R. MATHAR	6267	RSI/ IOTA														
		C. DE WIT	7055	MINIS	LASER RX (HP)													
		W. DE JONGHE	8555	PC														
		L. VERVAET	9118	TOOLS	(XPERT)													
5.8 Rédaction du rapport 'Résultats de test'						A.S.F.								v			X	5.8 Opstellen van het rapport 'testresultaten'

USER INFORMATICIAN

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRO-DUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CON-CERNED PER-SONS	COURSE	ORG.	REP.	S.D.M. FAZEN
		PERSONS	TEL.	GROUP														
6. CONVERSION ET MISE EN PRODUCTION							^							^				6. CONVERSIE EN INVOERING
6.1 Elaborer un plan détaillé de conversion et de mise en production					Super Project			X			X	X	Install.			X		6.1 Uitwerken van een gedetailleerd conversie- en invoeringsplan
		C. DE WIT	7055	MINIS														
6.2 Formation du personnel informatique		M. DE SMEDT	7145	PROD.												X		6.2 Opleiden van het computerpersoneel
	Operatortraining	C. DE WIT	7055	MINIS														
6.3 Elaboration d instructions de conversion et de mise en productio		B. BREMS Preparators	6292	PREP.	JCL REXX		CH-					X						6.3 Uitwerken van de instructies inzake conversie en invoering
		J-P MELEBEEK	6903	PROD.														
6.4 Conversion des données																		6.4 Converteren van de gegevens
6.5 Donner des renseignements sur le nouveau système																X		
6.5 Voorlichting over het nieuwe systeem geven																		
6.6 Formation des utilisateurs	Office Automation															X		6.6 Opleiding van de gebruikers
6.7 Formation du groupe de maintenance																X		6.7 Opleiding van het onderhoudsteam
6.8 Mise en production du nouveau système et son transfert		DBA's	7348 8860	IMS	IMS Compo-nents													6.8 Invoeren van het nieuwe systeem en het overdragen
		F. DE BRABANTER	7751	DB2														
		IMS/DB2 Prep.		PROD.	RE-COVERY FOR-MATS													
		W. DE JONGHE	8555	PC														
		R. MATTHAR	6267	RSI/ IOTA														
		J. VAN DE WATER	8715	PROD.	D.B.R.C.													
		EDP-Security (D. HARDY)	8503	SYS-TEMS			v	X					SEC-	v				

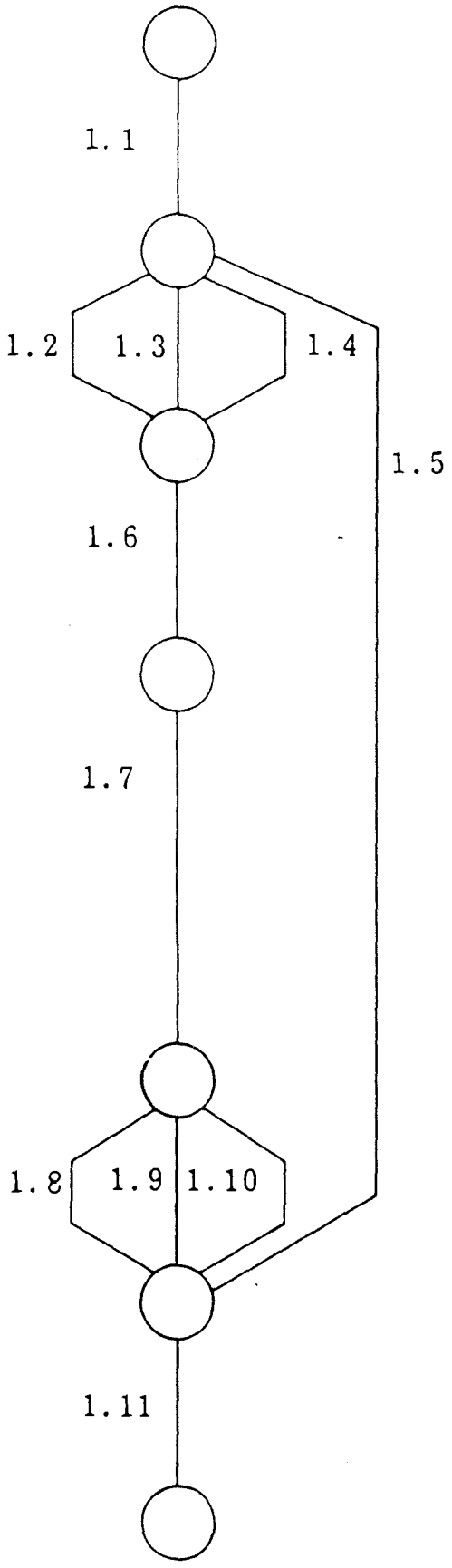
PREPARATION — PRODOS — PRODUCTION

PHASES S.D.M.	REMARKS	CONTACTS			TOOLS PRODUCTS	TECHNIQUES	DICT. (D.M.R.) CASE	SEC.	H.W.	S.W.	TEST	CONV.	FORM.	CONCERNED PERSONS	COURSE	ORG.	REP.	S.D.M. FAZEN
		PERSONS	TEL.	GROUP														
7. GESTION ET MAINTENANCE DU SYSTEME							^							^				7. ONDERHOUD EN BEHEER VAN HET SYSTEEM
7.1 Elaboration et utilisation du système de signalement des erreurs		B. BREMS	6292	PREP.	BETA91													7.1 Bouwen en in gebruik nemen van een systeem dat fouten signaleert
7.2 Planification périodique de maintenance		H. MARIEN	7348	IMS														7.2 Planning van de periodieke onderhoud
		P. FROIDCOEUR	7368	PROD.														
		F. DE BRABANTER	7751	DB2														
		P. DEVIGNE	7368	PROD.														
		C. DE WIT	7055	MINIS	BACK UP 3000													
7.3 Planification du traitement informatique		B. BREMS	6292	PREP.	CA-7 CA-11													7.3 Plannen van de computerverwerkingen
7.4 Prévention et restauration des fautes et perturbations		H. MARIEN	7348	IMS														7.4 Voorkomen en herstellen van fouten en storingen
		P. FROIDCOEUR	7368	PROD.														
		F. DE BRABANTER	7751	DB2														
		P. DEVIGNE	7368	PROD.														
		Dispatching	8833(F) 8822(N)	RSI/ IOTA														
		W. DOOREMONT	7197	RSI/ IOTA														
		P. HERMAN	6452	Techn. Inst.	NETMAN													
		C. DE WIT	7055	MINIS														
7.5 Surveillance des dispositions de sécurité	Disaster Recovery	EDP-Security (D. HARDY)	8503	SYSTEMS	R.A.C.F			X										7.5 Toezicht op de beveiligingsvoorzieningen
		DBA's	7348 8860	IMS														
		F. DE BRABANTER	7751	DB2	D.B.M.S. D.B.R.C.													
7.6 Modifications et mise à jour de la documentation. Réponse aux besoins en information ad-hoc		R. PARIZEL	6264	INFO-CENTRE	QUERY Lang. ICI	Documentation Standards									QUERY Lang.			7.6 Wijzingen en bijwerken van de documentatie. Voorzien in de ah-hoc informatiebehoeften
7.7 S'occuper des formations complémentaires																X		7.7 Verzorgen van de aanvullende opleidingen
7.8 Gestion des données et des fichiers		DBA's	7348 8860	IMS	LOAD UN-LOAD													7.8 Beheren van de gegevensbanken en de bestanden
		P. FROIDCOEUR	7368	PROD.														
		P. DEVIGNE	7368	PROD.														
	Capacity Planning	J. VAN DE WATER	5715	PROD.														
7.9 Evaluation du système et plans d'action		C. DE WIT	7055	MINIS	LASER RX (HP)													7.9 Evalueren van het systeem en maken van aktieplannen
		P. TATON	7323	Costs-Centre														
		J. MATTHIJS	7247	AUDIT														
		C. GRAAS	7037	DSI-M														
		M. COPPIETERS	6220	Manag.			v							v				

PRODUCTION / PREPARATION

ANNEXE3

MOGELIJKE PLANNING VAN DE ACTIVITEITEN VAN FASE 1



1.1 Definieer het probleem en de omvang van de definitie-studie.

1.2 + 1.3 Verzamel gegevens over de bestaande situatie en analyseer deze.

1.4 Bepaal doeleinden en eisen nieuwe systeem.

1.5 Leg vast nog op te lossen punten en van welke veronderstelling wordt uitgegaan.

1.6 Maak systeemschets.

1.7 Bepaal mogelijke hulpmiddelen en oplossingen.

1.8 Evalueer oplossingen en selecteer.

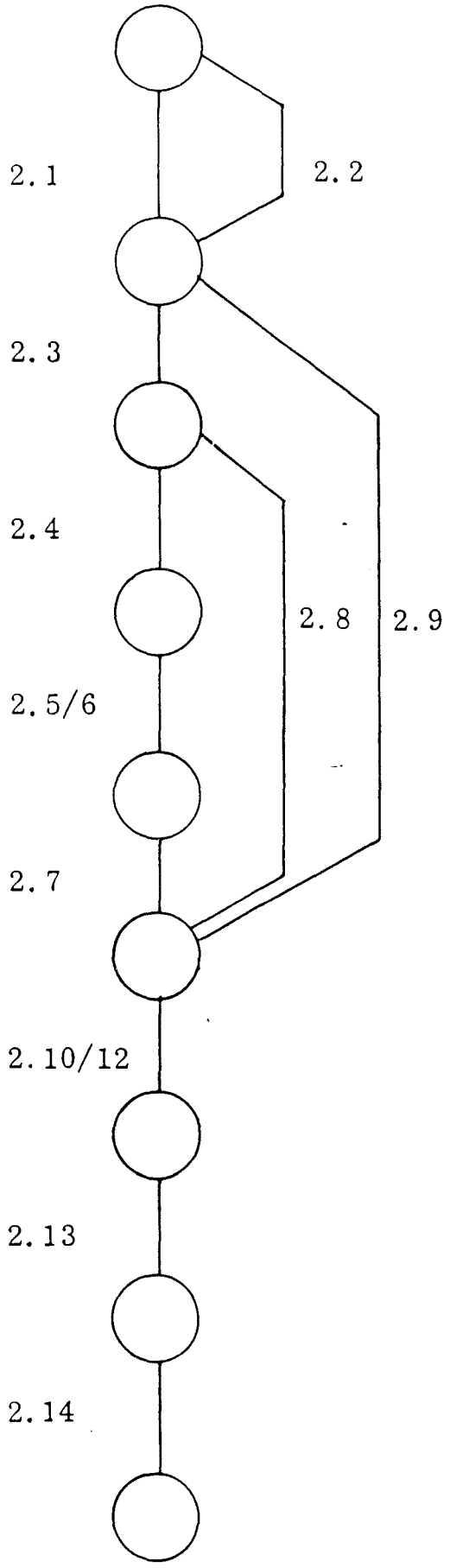
1.9 Bepaal de conversie en invoeringsproblemen.
Stel acceptatie criteria vast.

1.10 Maak een overall plan en een kosten/baten-overzicht.

1.11 Stel het definitie-studie-rapport samen.

N.B. De activiteiten 1.2 t/m 1.4 en 1.8 t/m 1.10 vormen dikwijls een iteratief proces.

MOGELIJKE PLANNING VAN DE ACTIVITEITEN VAN FASE 2



2.1 Specificeer systeem-eisen inclusief toekomstige.

2.2 Bepaal het raamwerk, waarbinnen het nieuwe systeem zal moeten functioneren.

2.3 Verdeel het systeem in subsystemen en beschrijf deze.

2.4 Bepaal de in- en uitvoer per sub-systeem en de onderlinge raakvlakken.

2.5/6 Maak processchema's en procesbeschrijvingen.

2.7 Specificeer eisen op het gebied van beveiliging en privacy.

2.8 Identificeer problemen op het menselijke vlak en oplossingen.

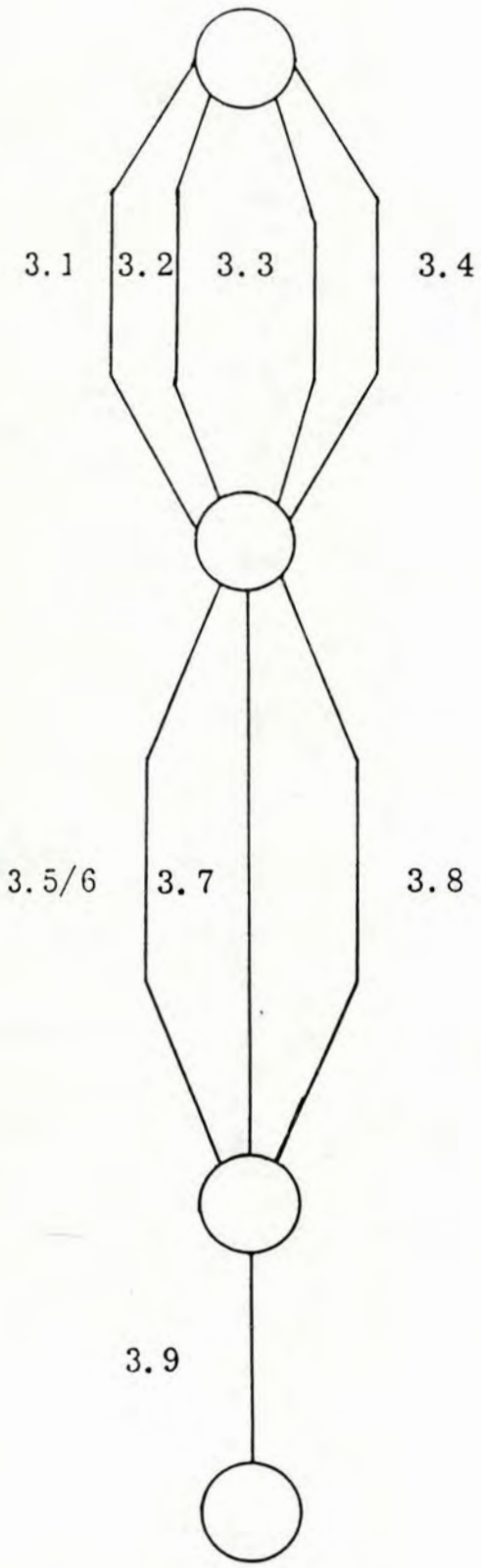
2.9 Ontwerp de gegevensstructuur.

2.10/12 Specificeer de benodigde faciliteiten op het gebied van:
- . data communicatie
- . apparatuur
- . programmatuur.

2.13 Maak een plan voor verdere ontwikkeling en invoering.

2.14 Stel het rapport "Functioneel ontwerp" samen.

MOGELIJKE PLANNING VAN DE ACTIVITEITEN VAN FASE 3



3.1 Ontwerp de procedures.

3.2 Ontwerp formulieren en alle computer in- en uitvoer.

3.3 Ontwerp de opslagstructuur.

3.4 Ontwerp veiligheidsmaatregelen.

3.5/6 Maak programma-specificaties en schema's.

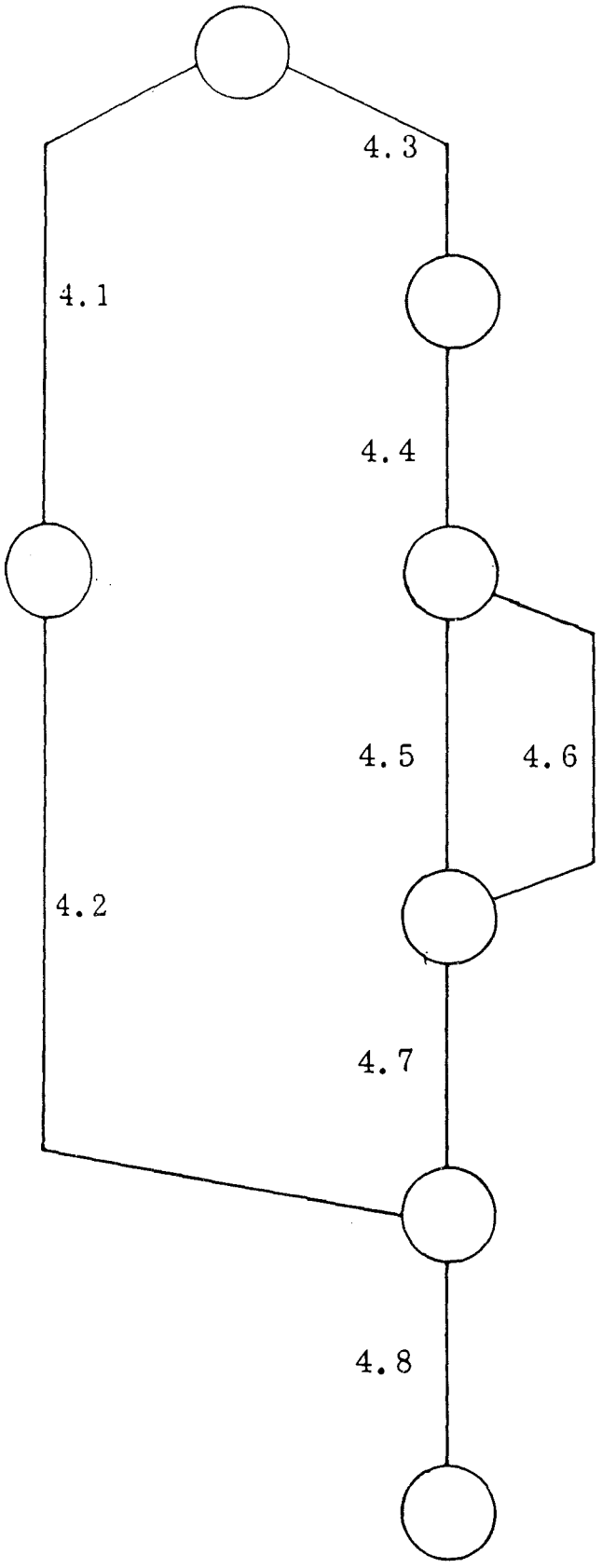
3.7 Specificeer de te gebruiken standaardprogrammatuur.

3.8 Maak een gedetailleerd programmeer- en testplan.

3.9 Stel het rapport "Technisch ontwerp" samen.

N.B. Fase 3 wordt dikwijls in combinatie met de fasen 4 en 5 gepland en uitgevoerd.

MOGELIJKE PLANNING VAN DE ACTIVITEITEN VAN FASE 4



4.1 Maak taakbeschrijvingen.

4.2 Stel de eisen vast waaraan het personeel en de werkomgeving moeten voldoen.

4.3 Detailleer de programmaspecificaties.

4.4 Codeer de programma's.

4.5 Draag zorg voor vertaling en corrigeer.

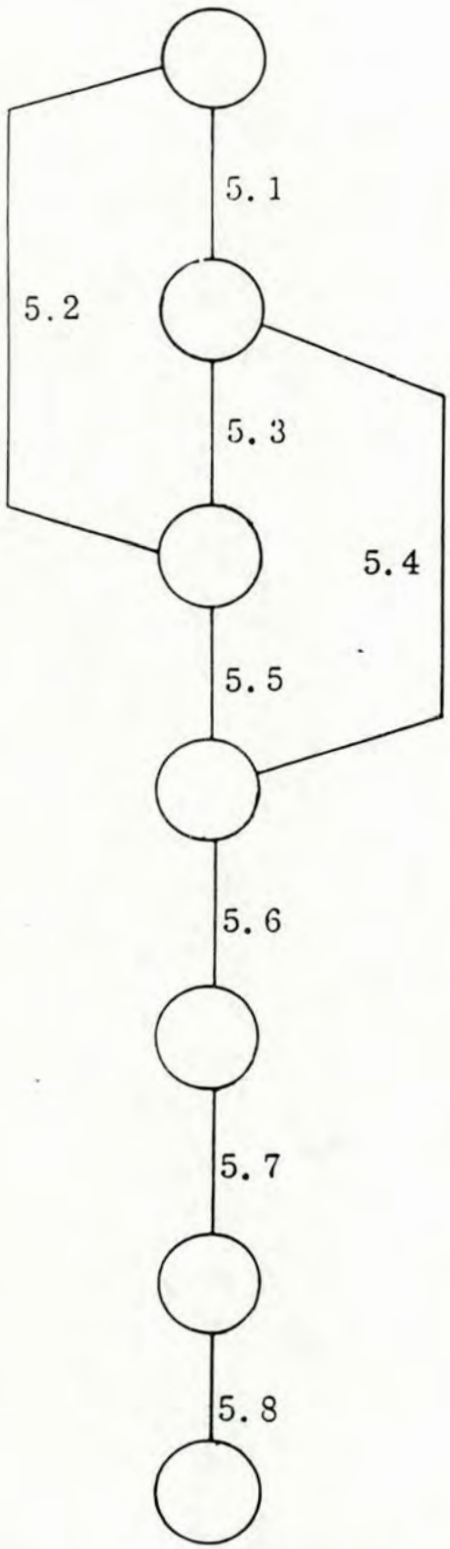
4.6 Maak programma-testgegevens.

4.7 Test de programma's.

4.8 Completeer de programmadocumentatie.

N.B. De activiteiten 4.1 en 4.2 worden ook dikwijls in fase 6 gepland.
De activiteiten 4.3 t/m 4.7 worden z.n. per programma afzonderlijk gepland.

MOGELIJKE PLANNING VAN DE ACTIVITEITEN VAN FASE 5



5.1 Maak een gedetailleerd testplan.

5.2 Installeer apparatuur, bedrijfsprogrammatuur en maak werkomgeving gereed.

5.3 Stel de verwerkingseenheden (jobs) vast en de onderlinge volgorde (job stream).

5.4 Test opleidingen, hulpmiddelen en procedures.

5.5 Stel de systeem- en acceptatietestgegevens samen

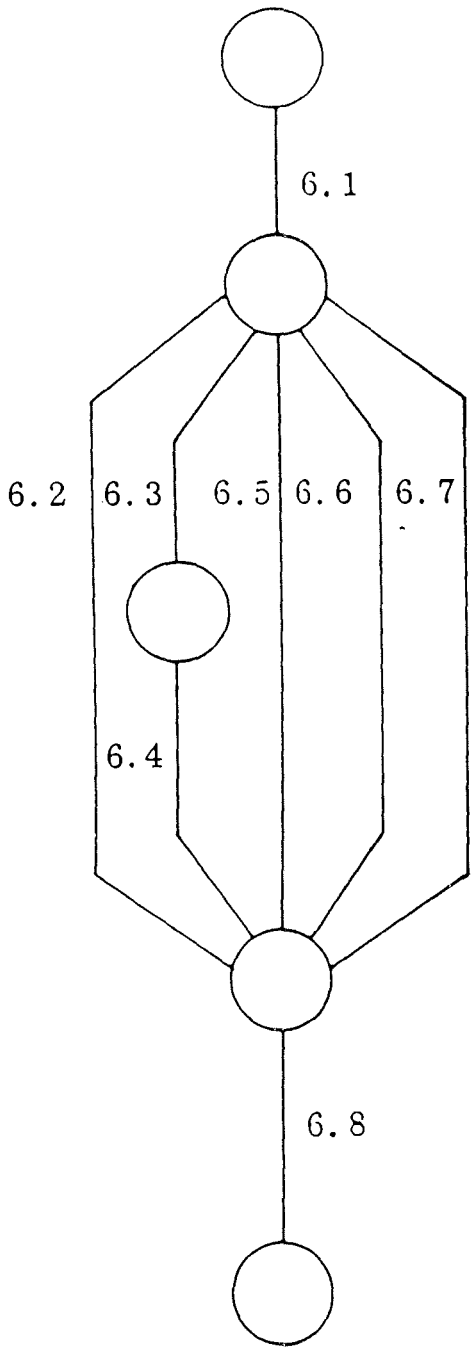
5.6 Test het (sub-)systeem.

5.7 Voer de acceptatietest uit.

5.8 Stel het rapport "testresultaten" samen.

N.B. Act. 5.2 zal in de regel reeds veel eerder zijn gerealiseerd. Dit is echter het laatst mogelijke tijdstip van uitvoering om het nog mogelijk te maken dat het testen geschiedt m.b.v. de apparatuur en programmatuur, waarop het systeem operationeel zal zijn. Indien (ook) in de gebruikersomgeving apparatuur e.d. geïnstalleerd moet worden, zal dat in de regel in het kader van fase 6 gepland worden (zie 6.0).

MOGELIJKE PLANNING VAN DE ACTIVITEITEN VAN FASE 6



6.1 Maak een gedetailleerd conversie- en invoeringsplan.

6.2 Leid het computerpersoneel op.

6.3 Maak instructies voor de conversie en de invoering.

6.4 Converteer de gegevens.

6.5 Geef voorlichting over het nieuwe systeem.

6.6 Leid het gebruikerspersoneel op.

6.7 Leid de onderhoudsgroep op.

6.8 Voer het nieuwe systeem in en draag het over.

N.B. In het kader van fase 6 worden dikwijls de volgende activiteiten uitgevoerd:

- 4.1 Maak taakbeschrijvingen
- 4.2 Stel eisen vast waaraan personeel en de werkomgeving moeten voldoen
- 5.3 Installeer apparatuur, bedrijfsprogrammatuur en maak werkomgeving gereed

Zie ook 6.0 "Gereedmaken werkomgeving en organisatie.

Réalisation, impression et reliures : SPRL Centre de Reprographie - Tél. : 081/22.87.72