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

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Gestion de prototypage DSL/Proto sous une base de données relationnelle

Penninckx, Dominique

Award date:
1986

Awarding institution:
Universite de Namur

Link to publication

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

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

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

Download date: 17. Apr. 2024

FACULTÉS
UNIVERSITAIRES
N.D. DE LA PAIX
NAMUR

INSTITUT D'INFORMATIQUE

GESTION DU SYSTEME DE PROTOTYPAGE DSL/PROTO SOUS UNE BASE DE DONNEES RELATIONNELLE

Directeur : François Bodart

Mémoire présenté par
Dominique PENNINCKX
en vue de l'obtention
du titre de
Licencié et Maitre
en Informatique

Année Académique 1985 - 1986

Remerciements

Je tiens à remercier tous ceux qui ont collaboré à la réalisation de ce mémoire. Tout d'abord, Monsieur François Bodart, promoteur du mémoire qui m'a proposé un stage chez Digital Equipment International à Genève. J'ai eu l'occasion de travailler au sein du groupe Application Development, sous la direction de Radu-Anton Eftimie. Je remercie pour leur aide les membres de l'équipe Methods and Tools, Janine Lambert, Patrick Scherrer et Geoffrey Darnton qui a supervisé mon travail pendant toute la durée de mon stage.

Je remercie également Anne-Marie Hennebert et Jean-Marie Leheureux pour leur collaboration constructive à la rédaction de ce mémoire.

Enfin, que tous ceux qui ont contribué d'une manière ou d'une autre à ce mémoire soient remerciés.

INTRODUCTION

INTRODUCTION

L'importance attachée à l'élaboration de maquettes et de prototypes est vraisemblablement liée à la conception d'objets complexes. Depuis bien longtemps, ces termes font partie du vocabulaire de base des ingénieurs, architectes ou constructeurs. Le prototype évoque le premier exemplaire d'un modèle, construit préalablement à une fabrication de série, tandis qu'une maquette évoque une esquisse, une ébauche ou une reproduction à échelle réduite ou de grandeur nature destinée aux études de prototypes.

Ces termes n'ont été introduits que récemment dans le monde du logiciel. L'expérience, souvent coûteuse, de la construction de produits logiciels a provoqué l'apparition de nombreuses démarches et techniques pour la conduite de la construction de logiciels. Une amélioration de la maitrise des projets logiciels en est incontestablement résultée, mais elle est jugée encore insuffisante. Dans le but d'améliorer la démarche de conception et de fabrication du logiciel, les approches classiques sont remises en cause; en particulier, il est proposé qu'une ou plusieurs étapes de maquettage et de prototypage fassent formellement partie du processus de conception du logiciel.

Ces approches, maquettage et prototypage, sont prometteuses, mais de nombreuses questions subsistent.
Que recouvrent les termes maquette et prototype?
Quelles en sont les acceptations?
Quels sont les approches et les outils utilisables?
Comment se situent-ils par rapport aux approches et outils classiques?
Quelle est la portée d'une maquette ou d'un prototype?
Comment maquettage et prototypage s'insèrent-ils dans le cycle de vie?
Quels sont les domaines d'application?

Le propos de ce mémoire n'est pas de fournir une réponse à ces questions, mais de mettre l'accent sur l'importance des outils de maquettage et prototypage en présentant une extension d'un système de prototypage

particulier, le système DSL-PROTO. Cette extension est la gestion du système sous une base de données relationnelle et la prise en charge par le système de la notion de transaction.

De nombreux produits de maquettage et prototypage existent sur le marché. Dans le but de répondre aux besoins et aux exigences des utilisateurs et des concepteurs, et de leur fournir des outils d'aide à la conception de logiciels toujours plus performants, ces outils sont en évolution constante.
L'évolution des outils et des techniques disponibles, la prise de conscience de l'importnce des étapes d'analyse et de spécification dans le processus de développement de logiciels, ont fortement influencés la recherche et la mise en oeuvre d'outils d'aide à la conception plus performants.

C'est dans cette optique que s'inscrit le propos de ce mémoire.
Dans le but de fournir aux utilisateurs et aux concepteurs un environnement de travail complet, pour le développement de systèmes d'information, le système d'aide à la conception IDA (Interactive Design Approach) offre deux outils de prototypage, le système DSL-SIM qui permet le prototypage des performances, et le système DSL-PROTO qui permet le prototypage des spécifications et du comportement du système d'information.

Plusieurs objectifs sont assignés au système de prototypage DSL-PROTO.
Premièrement, donner aux utilisateurs une vue globale du comportement général du système d'information.
Deuxièmement, valider et améliorer expérimentalement les spécifications du système pendant sa conception.
Troisièmement, se présenter comme un outil d'apprentissage des règles d'utilisation du système futur.
Actuellement, le système de prototypage se base sur la création et l'exécution d'un modèle réduit, d'une maquette du système d'information. Cette exécution à échelle réduite implique un certain nombre de contraintes, à savoir la gestion d'un volume réduit de données en mémoire centrale, l'exécution dans un environnment mono-utilisateur, la non considération des problèmes de sécurité et de fiabilité à l'exécution et l'ignorance des performances d'exploitation.
La réalisation des objectifs assignés au système implique l'évolution vers un système de prototypage basé sur la génération automatique et l'expérimentation d'un prototype du système d'information. La transformation du système de gestion des données en mémoire centraleen système de gestion de base de données relationnelle et la prise en charge de la notion de transaction vont permettre de lever les contraintes actuellement imposées au système, et vont contribuer à la réalisation des objectifs du système de

prototypage DSL-PROTO.

L'objet de la première partie de ce mémoire est de présenter le prototypage dans la conception et le développement des systèmes d'information dans les organisations, de montrer les limites des méthodes traditionnelles de développement, et de présenter le prototypage comme une solution alternative.
Nous présenterons le système d'aide à la conception de systèmes d'information IDA (Interactive Design Approach) qui offre deux outils de prototypage, le système DSL-SIM qui permet le prototypage des performances, et le système DSL-PROTO qui permet le prototypage des spécifications et du comportement du système d'information.
Nous analyserons plus particulièrement le système DSL-PROTO et proposerons un certain nombre d'évolutions et d'extensions possibles.

La deuxième partie de ce mémoire présente de manière approffondie deux évolutions du système, à savoir la gestion sous une base de données relationnelle et l'introduction de la notion de transaction.

Nous tenterons de montrer que l'introduction dans le système d'une base de données relationnelle et d'un module de gestion, ainsi que la prise en charge de la notion de transaction permettent de lever les contraintes imposées par l'exécution d'une maquette du système et de considérer le système de prototypage DSL-PROTO comme un vrai système d'exécution de spécifications, basé sur la génération rapide et automatique d'un prototype du système et sur l'expérimentation et la validation du comportement et des spécifications au début du processus de développement. Nous montrerons également que la gestion du système de prototypage sous une base de données relationnelle est un premier pas vers un système de génération de code final.

PREMIERE PARTIE

LE PROTOTYPAGE DANS LA CONCEPTION DES SYSTEMES D'INFORMATION

CHAPITRE 1 : PRESENTATION DU PROTOTYPAGE

1.1 Introduction et définitions

1.1.1 Motivations du prototypage

Le développement de logiciels est simplement le processus de préparation d'une version opérationnelle d'un système logiciel. Un bon développement requiert une approche rigoureuse, c'est-à-dire une méthode, qui est basée sur un ensemble cohérent de principes, de pratiques et de procédures, c'est-à-dire une méthodologie.
La plupart des méthodes actuelles, et leur méthodologie correspondante, sont basées sur la création d'une version du système par la production d'un ensemble de produits de travail, tels que les spécifications, la conception et l'implémentation [RIDDLE].

Le succès du développement d'application est lié aux aspects d'identification, de spécification et de validation des besoins qui doivent être satisfaits par le produit final. L'aspect spécification nécessite de la précision. Pour obtenir des spécifications précises et correctes du point de vue des besoins de l'utilisateurs, l'utilisation d'un prototype permet de démontrer les propriétés fonctionnelles du produit final. Utilisés comme outils de communication, les prototypes permettent aux utilisateurs de comprendre les spécifications du système bien mieux que n'importe quels textes écrits [MAYR].

1.1.1.1 Les limites des méthodes traditionnelles

L'approche traditionnelle au développement de logiciels est basée sur des spécifications écrites, gelées au début du processus de développement. Cette approche ne contient pas de processus de "feedback" d'une étape de développement vers les précédentes [NOSCK]. C'est donc une approche linéaire.

Cette approche traditionnelle est basée sur l'hypothèse que pendant l'étape d'analyse du système, il est possible de définir tous les besoins et les attentes des utilisateurs à un degré suffisamment précis pour que le système puisse être concu, construit et exécuté sans modification importante par la suite. La seconde hypothèse est qu'il est possible de prévoir toutes les conséquences positives et négatives de l'introduction du nouveau système dans l'organisation avant de commencer son développement [RZEVSKI].

L'efficacité de ces techniques d'analyse et de spécification qui se basent en premier lieu sur l'obtention de spécifications correctes est limitée, parce que les utilisateurs sont limités dans leur capacité de spécifier correctement leurs besoins [HOLLWAG].
En effet :

- Il leur est pratiquement impossible de spécifier à l'avance les critères d'évaluation du système. Il leur est plus facile de détecter les erreurs du système et de juger de son acceptabilité.

- Au moins les processus sont compris et au moins leurs critères d'évaluation sont définis, au plus il est difficile de spécifier à l'avance les besoins.

- Des changements dans les prévisions de l'organisation sont particulièrement significatifs de changements des spécifications du système.

- Plusieurs utilisateurs interagissant sur le sytème d'information rend la spécification encore plus difficile.

Ces aspects rendent la spécification et le développement des systèmes très difficiles et font que les méthodes traditionnelles de développement ne sont pas les plus adéquates et les plus satisfaisantes.

Dans l'approche traditionnelle au développement de systèmes, le cycle de vie peut se présenter de la manière suivante [BALLY]:

1. conception initiale
2. étude et analyse du problème
3. définition des spécifications
4. analyse détaillée
5. 'design'
6. implementation
7. tests
8. acceptation du système
9. opération et 'maintenance'
10. implémentation après analyse
11. modification.

Dans cette approche linéaire, chaque étape correspondant à une activité précise suit logiquement son prédécesseur, chaque activité étant terminée avant que la suivante ne commence. Cette approche linéaire n'est pas entièrement satisfaisante car l'environnemet changeant, les spécifications changent également. Et ce particulièrement pour les projets ayant un cycle de vie assez long, le système final ne correspond plus aux spécifications.
Le résultat est que l'utilisateur final n'accepte pas le système final après une coûteuse et longue période de développement.

1.1.1.2 La solution du prototypage

Le prototypage peut être vu comme une stratégie alternative pour surmonter les limites des méthodes traditionnelles.
Il devrait donc permettre à l'utilisateur d'acquérir de l'expérience pendant la période de développement, de déterminer rapidemment l'impact des spécifications sur l'organisation et de réduire les risques qui peuvent apparaitre pendant le processus de développement. En effet, il apparait important lors de la réalisation des projets de tester les spécifications du système en développement, car ceci permet de détecter et corriger les erreurs de conception, ainsi que de tester la spécification des besoins. La différence principale entre l'approche linéaire et l'approche prototypage est qu'un retour est possible de l'étape 'Opération' vers l'étape 'Définition des spécifications'. Ce procédé est utilisé pour améliorer les spécifications [HOLLWAG].

1.1.2 Définition

1.1.2.1 Le terme 'prototytage'

Dans le contexte logiciel, le prototype d'un produit est un modèle qui sacrifie à la précision dans certains domaines pour permettre une vérification rapide des fonctionnalités du produit. Un prototype est donc une forme de maquette exécutable où l'accent est mis sur certains aspects du produit, et où d'autres aspects sont ignorés, non décrits ou esquissés. Les prétentions du prototypage sont donc:

- d'être une base simple et efficace de compréhension entre l'utilisateur du logiciel et son concepteur,
- d'être une bonne introduction aux systèmes automatisés pour amener l'utilisateur à se familiariser avec son nouvel environnement de travail,
- et d'être une base d'amélioration des règles de traitements après analyse des prototypes.

Le prototypage permet donc le développement de logiciels au moyen

- d'interfaces utilisateurs expérimentaux,
- de spécifications définitivement fixées entre la fin du processus de spécification et l'introduction du logiciel chez l'utilisateur,
- d'outils de conception et d'implémentation qui permettent de réduire le temps entre ces deux processus.

Ces moyens permettent d'obtenir des logiciels acceptables.

1.1.2.2 L'importance du prototypage

Le sujet "développement de systèmes et prototypage" engendre la question suivante : dans quelle limite le prototypage est-il pertinent et applicable dans le développement de systèmes d'information dans une organisation? Cette question peut être envisagée de plusieurs points de vue [MATHIASS]:

- Du point de vue de l'idéaliste : Dans quelles situations le prototypage doit-il être utilisé?
- Du point de vue du réaliste : Quelles sont les conditions techniques et organisationnelles pour un prototypage réussi?
- Du point de vue du gestionnaire : Comment et dans quelles limites peut-on contrôler un projet basé sur une stratégie expérimentale?

1.1.2.2.1 Point de vue idéaliste

Le prototypage devrait être utilisé lorsque de nouvelles idées doivent être testées, lorsque de nouvelles technologies sont utilisées, ou lorsque de nouveaux environnements organisationnels sont créés. Le prototypage peut amener à une meilleure compréhension du domaine d'application, et il peut aider les concepteurs à acquérir des vues du système avant sa construction proprement dite. Ceci est rendu possible en testant et en expérimentant la solution conceptuelle proposée.

1.1.2.2.2 Point de vue réaliste

Le prototypage est utilisé comme un processus d'apprentissage et n'est pas suffisamment robuste pour devenir partie intégrante du système final, sauf si la stratégie basée sur des versions du système est explicitement utilisée. D'autre part, il est essentiel de planifier et de se rendre compte quand arrêter la construction du prototype et commencer son évaluation.

Un certain nombre de conditions techniques doivent être satisfaites pour pouvoir développer un prototype. Les aspects techniques à considérer sont:

- disposer d'outils et d'un environnement de programmation flexibles et rapides,

- fournir un haut degré de liberté, par exemple, flexibilité dans la programmation et logiciels modulaires,

- possibilité d'effectuer des tests complets reproductibles.

Les conditions organisationnelles sont en général moins importantes, on peut néanmoins en citer quelques-unes :

- disponibilité des ressources adéquates,

- accord complet sur les engagements et les responsabilités de chacun,

- volonté d'apprentissage.

Par contre, les implications au niveau organisationnel sont plus importantes :

- quelle doit être la répartition du travail entre utilisateurs et concepteurs,

- que faire quand des conflits d'interet apparaissent en relation avec des questions de conception,

- faut-il un staff de personnes plus compétent que lorsqu'une stratégie traditionnelle est utilisée.

1.1.2.2.3 Point de vue gestionnaire

La stratégie de prototypage facilite le développement de produits de haute qualité. Le contrôle de l'expérimentation et la détermination de la structuration et de la gestion du processus nécessitent une formalisation de contrats et d'efforts communs lors du processus de développement. Chaque réunion doit voir présents les groupes participants et doit résulter en un contrat spécifiant:

- le résultat attendu de l'effort fourni,

- le moment de l'évaluation suivante,
- une spécification détaillée des besoins de l'activité suivante,
- la volonté de continuer l'effort de coopération.

1.1.3 Evaluation

1.1.3.1 Impacts du prototypage

Le prototypage permet l'introduction dans la méthodologie de développement de logiciels d'un élément de feedback et de communication. Il prend en compte les besoins de communication spécifiques, les capacités de travail et les ressources disponibles.
Le prototypage n'est pas en soi une méthode de développement applicable à l'ensemble du processus de développement de systèmes. Il ne prescrit pas une séquence d'étapes qui garantit qu'un système exécutable satisfaisant les spécifications sera produit à partir des concepts vagues fournis par l'utilisateur. Il doit en fait être considéré comme une procédure dans le développement de systèmes, procédure qui doit être combinée avec d'autres procédures. Il peut être considéré comme une méthodologie applicable aux étapes d'analyse et de spécifications des besoins dans le but de produire des spécifications satisfaisantes pour l'utilisateur et le concepteur, qui pourront être expérimentées et améliorées.
Le prototypage offre le moyen de démontrer rapidement une version du système, et de déterminer à partir de plusieurs versions de ce type ce que veut l'utilisateur et ce que les concepteurs peuvent faire [FLOYD].

On peut considérer les étapes suivantes lors du développement de systèmes [FLOYD]:

- analyse générale de l'activité,
- analyse de la situation de travail et construction d'un modèle en termes familiers à l'utilisateur,
- essai d'élaboration des spécifications des besoins et conception d'un système potentiel utilisant les éléments de base du modèle du champ d'application,
- en se basant sur le modèle du système, construction d'un prototype et évaluation par les utilisateurs,
- répétition de ces étapes jusqu'à ce que les révisions fournissent une version acceptable des spécifications,

- construction du système final de manière incrémentale, en utilisant le dernier prototype comme partie des spécifications de la version suivante à développer.

Il est important de noter que le prototypage n'augmente pas le coût de développement du logiciel, puisque l'investissement supplémentaire dû au prototypage permet de réduire la probabilité d'incompréhension fondamentale et coûteuse.
Le prototypage substitue l'attente du système final par un processus d'apprentissage et d'expérimentation pratique. Il est important de noter que le prototypage apparait seulement au centre du processus de développement. Il ne solutionne pas automatiquement les problèmes d'implémentation sociale et organisationnelle [FLOYD].

1.1.3.2 Evaluation méthodologique

Le prototypage est une méthode permettant la création rapide de systèmes de travail dans les premières étapes du processus de développement de systèmes d'information. Beaucoup de méthodologies, y compris le prototypage, sont utilisées pour améliorer l'efficacité de ce processus. Cela émerge probablement d'une poussée technologique qui a pour but d'augmenter la productivité. Par contre l'aspect application constitue une contrainte pour obtenir des produits de qualité, parce que [SOL]:

- la maintenance nécessite de plus en plus de ressources,
- les applications deviennent de plus en plus complexes,
- l'orientation utilisateur-final des systèmes automatisés et d'aide à la décision demande une efficacité de plus en plus importante du développement des systèmes d'information.

Il existe un nombre important de méthodologies de développement de systèmes d'information, mais on peut constater que très peu sont efficaces. En effet, développer des systèmes d'information est une tâche complexe. Traditionnellement, cette complexité était abordée par ~~des~~ des processus de développement linéaire. Il semble qu'il soit plus efficace de se baser sur un processus de développement des sytèmes d'information caractérisé par un processus de communication et de feedback d'une étape de développement vers les précédentes.

1.2 Aspects et niveaux de prototypage

1.2.1 Etapes du prototypage

Le processus de prototypage consiste en 4 étapes : la sélection fonctionnelle, la construction, l'évaluation et l'utilisation future [FLOYD].

1.2.1.1 Sélection fonctionnelle

La sélection fonctionnelle consiste à déterminer les fonctions à prototyper. Elle doit être basée sur des tâches de travail qui pourront servir de cas modèles pour la démonstration. La différence entre les fonctions du prototype et le produit final peut être de deux types :

- les fonctions implémentées du système sont fournies sous leur forme finale, mais seulement certaines fonctions sélectionnées sont fournies (il s'agit du prototypage vertical),
- les fonctions ne sont pas implémentées en détail comme dans le système final, elles sont utilisées comme démonstration, une partie de leur effet étant omis ou simulé.

1.2.1.2 Construction

La construction consiste en la création du prototype. L'effort fourni pour le construire doit normalement être nettement inférieur à l'effort nécessaire pour construire le système final. Certains aspects importants du système final tels que sécurité des données, efficacité et performance

peuvent être omis sauf si ce sont ces aspects qui font l'objet du prototype.

1.2.1.3 Evaluation

L'évaluation doit être considérée comme l'étape décisive du prototypage, puisqu'elle fournit le feedback pour les processus de développement suivants.

1.2.1.4 Utilisation future

Il existe plusieurs possibilités d'utilisation future. En fonction des expériences acquises par le prototypage et de l'environnement de travail disponible, le prototype peut servir comme outil d'apprentissage et jeté après usage, ou il peut être utilisé comme composant du système final.

1.2.2 Classes de prototypage

Suivant les objectifs à atteindre, on peut distinguer trois classes de prototypage [FLOYD][LEGEARD]:

- le prototypage exploratoire, basé sur la clarification des besoins et des caractéristiques souhaitées pour le système final, et sur la discussion des solutions alternatives,

- le prototypage expérimentatif, basé sur la détermination de l'adéquation d'une proposition de solution avant d'entamer l'implémentation du système final,

- le prototypage évolutif, basé sur l'adaptation graduelle du système aux changements des besoins qui ne peuvent pas être déterminés dans les premières étapes du développement.

La frontière entre les prototypages exploratoire et expérimental est en fait assez vague. Mais la distinction est utile pour clarifier la relation entre le prototypage et le processus de développement de logiciel en tant que tel. Pour chaque approche, nous présenterons brièvement la relation entre le prototypage et le système final, le domaine d'application du prototypage et son utilisation future.

1.2.2.1 Prototypage exploratoire

Cette approche est basée sur le problème de la communication entre le concepteur et les utilisateurs, principalement pendant les premières étapes du développement. Les concepteurs ont en général peu de connaissance du champ d'application, tandis que les utilisateurs n'ont pas idée de ce que l'ordinateur peut leur apporter. Dans cette situation, une démonstration des fonctions du système permet de clarifier les idées et promouvoir une coopération entre les deux parties.
Le prototypage exploratoire est compatible avec une approche au développement de systèmes orientée phase et s'applique principalement aux phases d'analyse des besoins, d'analyse fonctionnelle et de définition des spécifications. Ce type de prototypage est considéré comme une aide à établir les caractéristiques que pourra offrir le système final. Il ne doit pas se limiter à démontrer une solution particulière, mais doit mettre l'accent sur les caractéristiques des différentes solutions alternatives au problème. Un tel prototype est construit graduellement et est le résultat d'un processus de communication entre concepteur et utilisateur. Il est généralement jeté par la suite puisqu'il n'a pas de relation directe avec le système final [FLOYD]:

1.2.2.2 Prototypage expérimental

Le prototypage expérimental peut être considéré comme une mise en valeur des spécifications du système final. Une solution est proposée au problème du client et est évaluée par expérimentation. Plusieurs domaines d'application peuvent être envisagés, à savoir la transparence de

l'interface homme-machine, l'acceptabilité des performances du système projeté, ou la faisabilité de la solution proposée en fonction des ressources disponibles. Il est important de définir à l'avance la nature de l'expérimentation et la stratégie choisie car elles sont la base du feedback et de la communication. On peut envisager plusieurs types de prototypage suivant la stratégie choisie:

- Simulation fonctionnelle complète, basée sur un prototype comprenant toutes les fonctions du système qui seront disponibles à l'utilisateur. Il est basé sur des facilités d'implémentation et de modification plutôt que sur une recherche d'efficacité du système résultant.

- Simulation de l'interface humain, basée sur le prototypage de l'interface homme-machine.

- Programmation squelette, basée sur la présentation de la structure générale du système par des fonctions jugées pertinentes. Cela implique la conception de l'ensemble du système et une réduction du champ d'application des fonctions. Les fonctions implémentées doivent permettre à l'utilisateur de tester son outil de travail futur. Ce type de prototypage permet de simuler l'efficacité du système final et de juger de son acceptabilité.

- Construction machine de base, qui implique l'implémentation de fonctions de base du système qui seront disponibles à l'utilisateur.

- Simulation fonctionnelle partielle, basée sur le test d'une hypothèse de travail du système.

L'utilisation future de ce type de prototypage dépend de la stratégie choisie et des outils disponibles. Il est soit intégré dans le système final, soit jeté.

1.2.2.3 Prototypage évolutif

Cette approche du prototypage est la plus puissante mais la plus éloignée de la signification donnée au terme 'prototypage'. Cette approche ne devrait pas s'appeler prototypage, mais développement en versions. Elle est basée sur l'expérience que l'organisation dans laquelle se développe le système évolue sans cesse et que de nouveaux besoins apparaissent, que le système lui-même évolue par son utilisation et que de nouvelles fonctionnalités apparaissent.

Il faut donc une stratégie dynamique qui voit le produit comme une suite de versions, où chaque version est évaluée et sert comme prototype à la version suivante.

1.3 Techniques et outils de prototypage

1.3.1 Techniques de prototypage

Les techniques les plus importantes en relation avec le prototypage sont la conception modulaire, la conception du dialogue et la simulation [FLOYD].

1.3.1.1 Conception modulaire

La conception modulaire est une technique importante pour les stratégies de prototypage qui ont pour but d'incorporer le prototype dans le système final. Ces stratégies incluent la simulation de l'interface utilisateur, la programmation squelette et le développement incrémental du système. Dans tous les cas, la modularité facilite le remplacement graduel des composants du prototype par les composants du système final.

1.3.1.2 Conception du dialogue

La conception du dialogue permet de rendre l'interface utilisateur transparent et flexible. Pour l'évaluation de l'interface, il doit être possible de discuter des différents aspects à plusieurs niveaux. Ces aspects incluent la structure générale du dialogue, le choix des commandes, le "layout" des écrans, les manipulations d'erreurs et de cas spécifiques et la possibilité de restructurer le système si nécessaire.

1.3.1.3 Simulation

Chaque programme est une simulation d'une activité du monde réel. Ainsi la simulation est utilisée pour valider les traitements des données, les traitements complexes et la fiabilité du système. En tant que technique de prototypage, la simulation permet de fournir ces parties du système et de les démontrer dans leur forme finale. La simulation est indispensable dans une évaluation réaliste d'un système.

1.3.2 Outils de prototypage

Depuis le début de l'utilisation du prototypage, plusieurs méthodes et outils ont été proposés. Ceux-ci incluent les langages de haut-niveau, les Systèmes de Gestion de Bases de Données (SGBD), les systèmes de définition de dialogue, les langages de spécification et les systèmes d'exécution symbolique.

Ces outils permettent la construction rapide de versions d'un système opérationnel qui peuvent être démontrées. Mais ce sont des outils généraux et non spécifiques au domaine d'application. De plus ils offrent souvent des interfaces peu flexibles.

Le support le plus prometteur semble être un environnement de programmation offrant un ensemble d'outils de prototypage et de construction de systèmes finals avec des facilités de transition et de transformation automatique du prototype en système final.

Une alternative plus réaliste à court terme, est une boite à outils de prototypage qui peut être appliquée à des domaines d'application spécifiques. Une telle boite à outils doit comprendre un système de traitement de textes, un langage de haut niveau, un SGBD, des facilités de simulation, un module statistique, un système de gestion d'écrans et de fenêtres, et un système de compilation/interprètation [FLOYD].

1.4 BIBLIOGRAPHIE

- [FLOYD] Ch. Floyd :
  " A systematic look at prototyping "
  from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [BALLY] L. Bally, J. Brittan, K. Wagner :
  " A prototype approach to information systems, design and developpment "
  Information and Management. 1 , 1977.

- [MAYR] H.C. Mayr, M. Bever, P.C. Lockemann :
  " Prototyping Interactive Application Systems "
  from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [LEGEARD] B. Legeard :
  " Techniques et aspects du prototypage "
  Génie Logiciel numéro 3 : "Maquettage et prototypage: outils et techniques"

- [RIDDLE] W. E. Riddle :
  " Advancing the state of the art in software system prototyping "
  from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [NOSCK] J. T. Nosck :
  " Organisation design choices to facilitate evolutionary development of prototype information systems "
  from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [HOLLWAG] I. Hollinde, K.H. Wagner :
  " Experience of prototyping in command and control information systems "
  from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [RZEVSKI] G. Rzevski :
" Prototypes versus pilot systems: strategies for evolutionary information system development "
from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [MATHIASS] L. Mathiassen :
Summary of the working group " Systems development and prototyping "
from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

- [SOL] H. G. Sol :
" A methodological assessment "
from " Approaches to prototyping " , Proceedings on the working conference on prototyping, Namur October 1985.

CHAPITRE 2 : LE PROTOTYPAGE DANS IDA

2.1 Situation actuelle

2.1.1 Introduction

L'analyse au chapitre précédent montre qu'il est de plus en plus nécessaire d'introduire des spécifications plus précises du système d'information pendant sa conception et d'en évaluer l'impact sur l'organisation avant son implémentation.

Il est interessant d'analyser l'importance de la spécification des besoins du système d'information et de leur évaluation, du point de vue de la contribution du système d'information en développement à la réalisation des objectifs et à la performance de l'organisation. Cette approche a pour but d'améliorer l'efficacité et l'effectivité des systèmes d'information en vue de leur implémentation.

Dans la perspective de la réalisation de cet objectif, le système IDA (Interactive Design Approach) fournit un environnement de travail pour le développement de systèmes d'information dont les constituants sont les suivants [BOD85D] :

- des modèles et un langage associé pour une spécification rigoureuse des systèmes d'information;
- deux outils logiciels permettant l'évaluation expérimentale des spécifications, par prototypage et par simulation, avant leur implémentation.

2.1.2 Spécifications des besoins du système d'information

Le modèle de spécification des besoins sur lequel se base IDA s'appuie sur trois représentations du système d'information à développer [PIGNEUR] :

- la relation processus-événement qui spécifie la condition d'activation et l'enchainement des processus. Ce sous-modèle fournit des concepts et des mécanismes pour spécifier le comportement fonctionnel du système d'information.

- la relation classique processus-données qui spécifie les données, leur structure, leur utilisation et leur traduction par les processus. Ce sous-modèle utilise des concepts et des mécanismes pour spécifier la structure des données stockées dans la mémoire du sytème et véhiculées au moyen de messages, et l'utilisation et la traduction des données par les traitements.

- la relation processus-ressource qui spécifie les ressources humaines, techniques et organisationnelles utilisées par les processus. Ce sous-modèle permet de caractériser les ressources requises pour l'exécution des processus.

2.1.3 Evaluation de la spécification des besoins

Les outils logiciels fournis par IDA permettent l'expérimentation des besoins, c'est-à-dire l'expérimentation du comportement du système d'information avant son implémentation. Le contrôle direct de ces outils par les spécifications réduit la divergence entre le système spécifié et le système simulé ou prototypé. L'ensemble de ces outils réalise les fonctions suivantes [BOD85D] :

- la génération de rapports documentaires présentant de diverses manières les différents aspects du système d'information,

- l'exécution de tests de cohérence et de complétude des descriptions proposées,

- la génération et l'exécution de simulation des performances pour tester le caractère réalisable du système d'information en développement,

- la génération de prototypes fonctionnels pour tester l'effectivité des spécifications.

Les outils de simulation et de prototypage permettront de tester si les spécifications des besoins sont réalisables et conformes aux besoins dans le but de satisfaire des critères de cohérence et de complétude.

L'outil de simulation, correspondant à un outil de prototypage des performances, est un exemple typique d'outil de prototypage expérimental utilisant la stratégie de simulation fonctionnelle partielle (cfr 1.2.2.2.). En effet, le but de son utilisation est simplement de tester une hypothèse de travail faite sur le système, à savoir vérifier que la solution proposée produit des résultats acceptables dans la majorité des cas en utilisant le volume de ressources disponibles. L'outil peut également être considéré comme un exemple d'outil de prototypage expérimental basé sur la stratégie de programmation squelette. Cette forme de prototypage est interessante pour montrer l'efficience du système futur et démontrer l'interaction des tâches manuelles et automatisées.

L'outil de prototypage proposé par IDA peut être considéré tant comme outil de prototypage exploratoire que comme outil de prototypage expérimental (cfr 1.2.2.1. et 1.2.2.2.).
En effet, il peut être considéré comme outil de prototypage exploratoire en ce sens qu'il est d'abord un outil de prototypage fonctionnel. Il permet de démontrer les fonctions possibles du système, de montrer les solutions alternatives et de discuter leurs caractéristiques propres. Il est un outil de communication entre utilisateurs et concepteurs dans le but de déterminer les caractéristiques et les fonctions du système final.
L'outil de prototypage est également un exemple d'outil de prototypage expérimental en ce sens que la solution proposée aux utilisateurs peut être évaluée et validée de manière expérimentale, par exécution des spécifications des besoins définis par les utilisateurs.

2.2 Prototypage des performances

2.2.1 Introduction

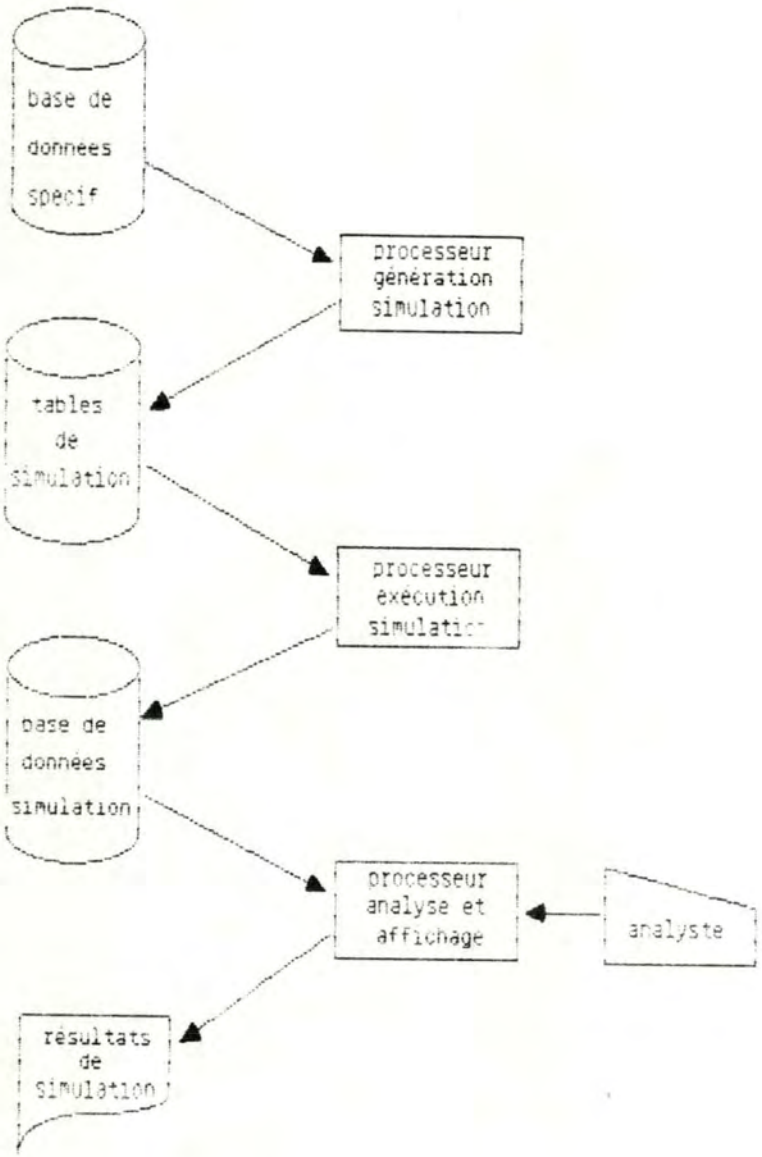
L'outil de simulation automatique DSL-SIM permet d'analyser l'efficience des spécifications du système d'information, c'est-à-dire d'analyser les performances du système à un coût donné, dans un environnement de ressources spécifiques. Il fournit un moyen d'analyse expérimentale du comportement du système d'information pendant un certain temps, et un moyen d'évaluation de ses performances avant son implémentation.

L'approche intégrée permet au concepteur et aux utilisateurs d'évaluer expérimentalement les performances du comportement du système d'information en cours de conception, d'évaluer les ressources requises pour produire ces performances et de détecter des fonctionnements erronés du système. [BOD85D] [BOD85J] [BLIN]

2.2.2 Utilisation de l'outil

Dans un premier temps, les spécifications sont exprimées en DSL (Dynamic Specification Language), enregistrées dans une base de données des spécifications et vérifiées statistiquement du point de vue cohérence et complétude. Ensuite, un modèle exécutable est produit automatiquement et exécuté pour simuler le comportement du système spécifié et produire les mesures de performance souhaitées.

Le schéma suivant présente l'architecture générale de DSL-SIM, qui consiste en trois processeurs.



- le processeur de génération de la simulation qui transforme automatiquement la partie concernée des spécifications en tables de simulation pour le processeur d'exécution de la simulation. Cette génération automatique empèche une divergence entre le système simulé et le système spécifié, et permet une analyse rapide des conséquences des changements dans la base de données des spécifications.

- le processeur d'exécution de la simulation conduit la simulation suivant les tables de simulation générées automatiquement par le processeur de génération de la simulation, enregistre les événements survenant pendant la simulation, rassemble les données statistiques sur ces survenances et les enregistre dans une base de données des résultats de la simulation.

- le processeur d'affichage et d'analyse de la simulation analyse les résultats statistiques de la simulation et produit des rapports de simulation.

2.2.3 Test de l'efficience du système

Pendant l'exécution d'une simulation de performance, un certain nombre de mesures sont enregistrées dans une base de données pour des analyses statistiques du comportement du système d'information en développement. Ces mesures peuvent être utilisées pour détecter les problèmes de performance et pour en localiser les causes [BOD85J] [BOD85D].

Les méthodes d'interprètation des résultats sont les suivantes [BOD85D] [BOD85J]:

- Une première étape consiste en l'identification et la localisation des divergences entre les performances attendues, spécifiées avant la simulation, et les performances calculées, établies par la simulation. Deux types de mesures peuvent être utilisées dans ce but:

  - une mesure de délai entre deux classes successives d'événements,

  - une mesure du nombre d'événements dans chaque classe, pour contrôler si les nombres d'événements des classes suivantes et précédentes concordent.

- En ce qui concerne les divergences entre les délais estimés et les délais calculés, les causes du manque de déclenchements de processus seront identifiées par l'analyse des mesures concernant les points de synchronisation. Pour un point de synchronisation où le temps de réalisation est grand, les événements contraignants contribuant au point de synchronisation seront identifiés. Pour ces événements, s'ils sont internes, les chemins dans lesquels ils surviennent seront analysés.

- L'étape suivante est la détection, principalement dans les chemins localisés à l'étape précédente, des classes de processus pour lesquels le temps d'attente est anormalement long et/ou les interruptions sont trop fréquentes ou trop longues. Ces situations seront découvertes par analyse des mesures concernant le comportement des processus.

- Finallement, les ressources critiques sont identifiées. Une ressource est critique si elle est la cause de temps d'attente et/ou d'interruption anormaux, soit à cause de sa trop forte utilisation, soit à cause de sa capacité disponible trop petite. D'un point de vue économique, une ressource peut aussi être critique quand sa disponibilité est trop importante par rapport à son niveau d'utilisation. Une ressource critique est détectée par l'analyse des mesures concernant le comportement des ressources.

Une solution effective est obtenue par un processus de recherche basé sur l'interprétation des résultats de la simulation.

2.3 Prototypage des règles de traitement

2.3.1 Introduction

L'outil de prototypage fonctionnel DSL-PROTO permet la dérivation de prototypes de système à petite échelle, sous la forme de code-à-jeter, dans le but de tester si les spécifications des besoins sont efficaces et conformes aux besoins de l'organisation. Un prototype permet aux utilisateurs d'exécuter les spécifications du système, pour les comprendre et les contrôler. Il leur permet de déterminer expérimentalement si elles produisent les résultats attendus [BOD85J] [BLIN].

2.3.2 Buts du système de prototypage

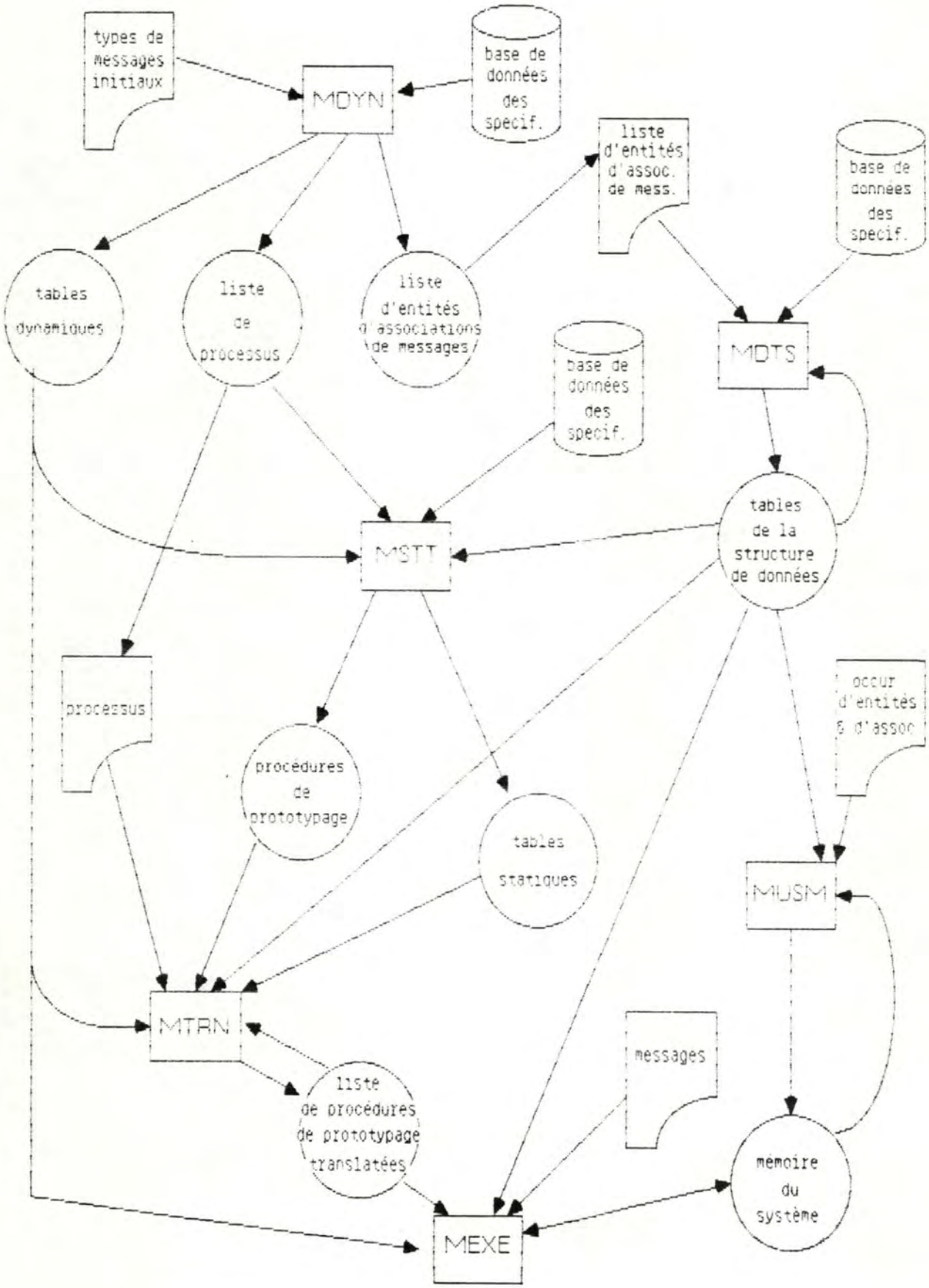
Trois buts précis peuvent être assignés au système de prototypage DSL-PROTO [BOD85D] :

- premièrement, fournir à l'utilisateur une vue globale du comportement du système d'information. DSL-PROTO permet le prototypage des fonctions du système d'application et des flux des messages, ainsi que des interactions entre les traitements manuels et automatiques.

- deuxièmement, permettre une validation expérimentale des spécifications du système d'information par les analystes et les utilisateurs dans le but de les comprendre, de les valider et de les améliorer.

- troisièmement, être utilisé comme outil d'apprentissage des règles d'utilisation du système d'information pendant son développement.

2.3.3 Présentation de l'architecture de DSL-PROTO



2.3.3.1 Description du rôle de chaque module

1. Module de génération de l'environnement dynamique (MDYN):

A partir d'une liste de messages initiaux qui déterminent le champ global, le module MDYN extrait de la base de données des spécifications les phrases DSL relatives à la description de la dynamique engendrée par ces messages initiaux et enregistre ces données dans les tables de la dynamique. Les différents processus rencontrés sont enregistrés dans une liste de processus, et les types d'objets référencés par ces processus sont enregistrés dans une liste. Le module procède à une série de contrôles de cohérence, par exemple:

- la détection des circuits dans les spécifications dynamiques;
- la vérification de la compatibilité des calendriers;
- la vérification de la définition des messages initiaux dans la base de données des spécifications;
- la vérification de la cohérence de la syntaxe des points de synchronisation avec la définition des processus qui y contribuent;
- la vérification de la cohérence entre la définition des attributs et les conditions de déclenchement; ...

2. Module de création des tables de définition (MDTS):

A partir d'une liste de types d'objets (provenant du module MDYN ou fournie par l'utilisateur), le module MDTS extrait de la base de données des spécifications la description de chacun des types d'objets contenu dans la liste pour construire les tables de définition de la structure de données. Le

module effectue une série de contrôle, par exemple:

- les contrôles de la forme canonique du schéma entité/association [BERTINCHAMPS].
- la vérification de la spécification du format de chaque type d'élément;
- la vérification de l'écriture du domaine de valeur d'un type d'élément ;
- la vérification de l'identification de chaque type d'entité;
- la vérification de l'existence d'un contenu pour chaque type d'entité et chaque type de message ;
- la vérification de l'unicité du nom de chaque type d'objet; ...

Les tables de définition peuvent être mises à jour à tout moment en fournissant au module une liste de types d'objets. Actuellement, cette mise à jour ne consiste qu'en un ajout de types d'objets. La modification ou la suppression d'un type d'objet interrompt l'exécution du module.

Le module ne contrôle pas la cohérence entre les descriptions déjà enregistrées dans les tables de définition et les descriptions contenues dans la base de données des spécifications. Les descriptions utilisées par les autres modules sont toujours celles enregistrées dans les tables de définition, et non celles enregistrées dans la base des spécifications. Une mise à jour de types d'objets dans la base de spécifications doit donc toujours être suivie d'une mise à jour des tables de définition.

3. Module de génération de l'environnement statique (MSTT):

A partir de chacun des types d'objets définis dans la table de définition, le module MSTT extrait pour chaque processus du champ global les phrases relatives à la relation statique entre ces processus et le type d'objet sélectionné, et construit les tables de la statique. Le module procède à l'extraction des procédures de prototypage associées à chacun des processus appartenant à une liste qui lui est fournie

par l'utilisateur, et vérifie que ces processus appartiennent à la liste des processus du champ global. Les procédures de prototypage extraites de la base des spécifications sont stockées dans un fichier séquentiel.

4. Module de mise à jour de la mémoire du système (MUSM):

La mémoire du système de prototypage est mise à jour par le module MUSM auquel l'utilisateur fournit des occurrences des types d'entités et d'associations qu'il désire introduire dans la mémoire.

Le module guide l'utilisateur pour l'enregistrement des occurrences. Pour chaque type d'objet pour lequel l'utilisateur veut enregistrer une occurrence, le module indique les types d'éléments et leur format associé, ainsi que les types d'objets (groupe ou élément) constituant l'identifiant pour lesquels l'utilisateur doit fournir une valeur.

Le module contrôle que les valeurs données par l'utilisateur aux types d'éléments respectent les formats spécifiés et vérifie que l'identifiant est unique.

Le module contrôle également les connectivités des occurrences des associations, les connectivités minimales sont vérifiées à la fin de chaque mise à jour de la mémoire du système et les connectivités maximales sont vérifiées pendant la mise à jour. Une connectivité minimale non respectée entraine un avertissement à l'utilisateur, une connectivité maximale 0-N ou 1-N avec N défini constant non respectée est simplement signalée à l'utilisateur tandis qu'une connectivité maximale 0-1 ou 1-1 non respectée n'est pas admise.

5. Module de traduction des procédures (MTRN):

A partir d'une liste de procédures de prototypage qui lui est fournie par l'utilisateur, le module MTRN procède à la traduction en un pseudo-langage de chacune

des instructions de la procédure de prototypage sélectionnée. Les instructions ainsi traduites sont stockées dans une table et pourront directement être exécutées par le module **MEXE**. Le module effectue certaines vérifications:

- il contrôle que les processus pour lesquels l'utilisateur demande une traduction de la procédure de prototypage associée appartiennent à la liste des processus du champ global;

- il effectue des contrôles syntaxiques et sémantiques des instructions de la procédure de prototypage;

- il contrôle la cohérence entre les instructions de la procédure et les phrases DSL exprimant la statique du processus considéré.

6. Module d'exécution du prototype (MEXE):

L'utilisateur fournit au module **MEXE** des occurrences des messages externes à partir desquels il désire que l'exécution se déroule. A partir de ces occurrences de messages externes, les tables de la dynamique déterminent les processus à exécuter, leur condition de déclenchement et leur synchronisation.

Le code des procédures correspondant à ces traitements est fourni par les tables des procédures en pseudo-langage. Les types d'objets manipulés par ces traitements sont décrits dans les tables de définition de la structure de données, les valeurs initiales des types d'entités et d'associations ont été enregistrées précédemment dans la mémoire du système. En cours d'exécution, les procédures mettent à jour la mémoire du système.

2.3.4 Utilisation du système

Réaliser le système d'information à petite échelle signifie que [BOD85D]:

- le système d'information ne contient qu'un volume réduit de données, c'est à dire un volume qui peut être stocké et géré en mémoire centrale,

- le prototype est exécuté dans un environnement local et mono-utilisateur, ce qui signifie que les problèmes de concurrence et de simultanéité ne sont pas pris en compte,

- le système de prototypage ne tient pas compte des problèmes de sécurité et de fiabilité pendant l'exécution, c'est à dire des problèmes de gestion des copies de sécurité et des points de reprise,

- le système de prototypage ne prend pas en considération la performance à l'exécution.

Dans le but de minimiser les discordances entre les spécifications et le prototype, un nombre maximum d'éléments du programme de prototypage est généré automatiquement à partir de la base de données des spécifications où les spécifications constituent des contraintes qui doivent être vérifiées par le programme de prototypage.

2.3.5 Présentation du langage de prototypage

Le langage DSL (Dynamic Specification Language) est un langage d'expression des spécifications d'un système d'information. Pour l'exécution de la maquette du système, un programme maquette doit être écrit. Ce programme est constitué de phrases DSL et de textes procéduraux emboités dans un description DSL. Les phrases DSL correspondent à un sous-ensemble des phrases DSL relatives à un sous-ensemble des objets DSL.

Il existe deux types de textes procéduraux emboitables dans un texte écrit en DSL:

- le premier type concerne la description des points de synchronisation,

- le deuxième type correspond à l'expression algorithmique de la procédure attachée à un processus.

2.3.5.1 Phrases DSL utilisables dans un programme maquette

Les types d'objets pris en compte par DSL-PROTO sont les objets de type association, condition, élément, entité, groupe, message, processus, point de synchronisation, paramètre du système et attribut. Pour ces types d'objets, DSL-PROTO prends en considération les aspects structure de données, statique, dynamique et architecture de système.

DSL-PROTO admet l'écriture de la dynamique soit par les processus, soit par les messages.

Exprimer la dynamique par les processus signifie que c'est l'exécution des processus qui guide l'exécution du prototype. C'est l'événement 'terminaison du processus' qui déclenche l'exécution du processus suivant dans le schéma de la dynamique. Les messages générés par le premier processus sont alors disponibles pour le processus suivant déclenché.

Par contre, l'expression de la dynamique par les messages signifie que ce sont les génération de messages qui constituent les événements déclencheurs qui guident l'exécution du prototype. En effet, l'idée est que comme la communication des données entre processus se fait obligatoirement par des messages, c'est la génération du message qui constitue l'événement qui déclenche l'exécution du processus suivant défini dans le schéma de la dynamique.

Les deux procédés sont actuellement acceptés par le système de prototypage, mais la description de la dynamique par les messages parait plus avantageuse parce qu'elle semble sémantiquement plus proche du comportement dynamique du système. Les règles et les modalités de prise en compte sont détaillée dans [RAPP].

2.3.5.2 Langage d'expression des points de synchronisation

Le langage d'expression des points de synchronisation est caractérisé par les opérandes qui sont soit des références aux contributions à ce point de synchronisation, soit des opérateurs.
Une opérande qui correspond à une référence à une contribution peut être la terminaison d'un processus,

activation d'un processus, la génération d'un message ou la réalisation d'un point de synchronisation.
Une contribution est caractérisée par l'événement qui contribue à ce point de synchronisation, par la condition et le sens du test associé à cette condition.
Toutes les contributions à un point de synchronisation doivent être référencéesune et une seule fois dans le prédicat de réalisation de ce point de synchronisation.
La plupart des conditions de synchronisation peuvent être exprimées à l'aide des opérateurs unaires et binaires suivants, AND, OR, SAME-ORIGIN, COUNT et GROUP-ALL.
Une description détaillée de ces opérateurs est présentée dans [MANREF].
L'introduction de la notion de procédure pour exprimer la réalisation des points de synchronisation permettrait de réaliser n'importe quel point de synchronisation.

2.3.5.3 Langage d'expression d'algorithmes

PROTO est un langage d'expression d'algorithmes de haut niveau dans lequel sont intégrées des primitives de manipulation des structures de données.

Le langage manipule les objets de type message, entité, association, groupe, élément, condition DSL, nom de procédure et variable de travail. Une spécification détaillée de ces objets est présentée dans [RAPP].

Le langage PROTO dispose des structures algorithmiques classiques, instruction d'assignation, de branchement, de boucle et instruction de modification de boucle, dont une spécification précise est présentée dans [RAPP].

Le langage permet l'introduction de la notion de procédure à l'intérieur de la procédure de prototypage dans le but de découper celle-ci en composants logiques. La syntaxe de déclaration et d'appel ainsi que les règles d'utilisation sont présentées dans [RAPP].

En ce qui concerne les primitives de manipulation des structures de données, elles permettent de manipuler ces structures représentées en mémoire centrale par des tables et correspondant aux messages, aux entités et aux associations. Le langage PROTO dispose des primitives de réception et de génération de messages, des primitives de création, de modification et de suppression d'entité ou

d'association, et d'une primitive d'affichage des données à l'écran. Une spécification détaillée de ces primitives est présentée dans [RAPP].

Le langage PROTO en tant que langage de programmation met à disposition de l'utilisateur des primitives de manipulation de données et des structures algorithmiques qui permettent d'écrire les procédures de prototypage associées aux processus du prototype.
Mais pour l'écriture de ces procédures, l'utilisateur doit tenir compte d'un certain nombre de restrictions imposées par le langage.

On peut citer comme exemples de restriction [RAPP]:

- deux blocs portant sur le même type d'objet ne peuvent pas être imbriqués,
- deux blocs ne peuvent pas se chevaucher,
- l'assignation d'un groupe à un groupe, d'un message ou d'une entité à un autre message ou entité n'est pas admise, ce qui alourdit l'écriture puisque l'assignation doit se faire de chaque élément du premier type d'objet à chaque élément du second type.

Pour éviter les restrictions qu'imposent un langage de programmation, les procédures de prototypage pourraient se présenter sous la forme d'une liste de requêtes correspondant aux traitements à effectuer sur les données.
L'analyse approfondie de cette évolution est présentée au chapitre 1 de la partie suivante.

2.3.6 Test de l'effectivité

Le test d'efficacité peut être fait de plusieurs manières [BOD85D]:

- Par les différents processeurs, DSL-PROTO contrôle la cohérence entre la spécification des besoins et leur implémentation.
- A l'exécution du prototype:

  Avant l'exécution, l'utilisateur peut spécifier le niveau de trace qu'il souhaite avoir pendant l'exécution et qui caractérise le niveau de détail des informations qui lui sont communiquées pendant l'exécution. Le système lui communique l'état de la mémoire du système, l'état des messages et des

événements suspendus.

Pendant l'exécution du prototype, pour chaque processus, le système communique à l'utilisateur les messages recus et générés par l'exécution et les actions effectuées sur les données détaillées suivant le niveau de trace spécifié.

A la fin de l'exécution du prototype, le système informe l'utilisateur de l'état de la mémoire du système, de l'état de l'ensemble des messages et des événements suspendus à ce moment.

- L'analyste ou l'utilisateur peut choisir des valeurs initiales à assigner aux messages externes dans le but d'exécuter un chemin bien défini.

- Dans un but d'apprentissage, l'utilisateur peut demander de sauver l'environnement relatif à une sous-partie du système dans le but de la réexécuter dans les mêmes conditions.

2.4 Critiques de l'outil DSL-PROTO

2.4.1 Introduction

L'évaluation du système de prototypage DSL-PROTO peut se baser sur plusieurs aspects.
Nous nous proposons de commencer par présenter brièvement les limites du domaine d'application du système. En effet, DSL-PROTO ne permet que le prototypage fonctionnel, et non pas le prototypage de l'interface homme-machine, ni le prototypage de l'implémentation du système.
Dans un premier point, l'évaluation portera sur l'analyse des personnes susceptibles d'être interessées et concernées par l'expérimentation d'un prototype du système. DSL-PROTO s'adresse aux utilisateurs organisationnels et aux informaticiens, plutôt qu'aux utilisateurs exploitant du système d'exploitation.
En second lieu, nous nous proposons d'évaluer les objectifs du système du point de vue de leur réalisation. Les objectifs sont-ils atteints, et dans le cas où ils ne le sont pas, peuvent-ils être réalisées ou doivent-ils être considérés comme irréalisables?
Dans un troisième point, nous examinerons les facilités de réalisation du prototype du point de vue du langage mis à disposition de l'utilisateur pour l'écriture des procédures de prototypage, ainsi que du point de vue des règles d'écriture et d'évaluation du prototype.
Dans un quatrième point, nous analyserons les possibilités d'utilisation future du prototype produit.

2.4.2 Critique du domaine d'application du prototypage

DSL-PROTO permet aux utilisateurs de prototyper des fonctions du système d'information en cours de développement. Il est principalement un outil de prototypage fonctionnel. Il n'est pas concu pour réaliser le prototypage de l'interface homme-machine ni le prototypage de l'imlémentation du système.

Le prototypage de l'interface homme-machine correspond au prototypage du dialogue entre l'utilisateur et la machine. Ce prototypage est basé sur la présentation à l'utilisateur du dialogue sous sa forme finale sans souci du comportement de toutes les autres parties du système.

Le système DSL-PROTO ne donne pas la possibilité de prototyper l'implémentation du système. Un système de prototypage de l'implémentation nécessite l'implémentation complète de certaines fonctions ou primitives de base du système, ainsi que de leurs interfaces avec les fonctions de plus haut niveau. Cela permet à l'utilisateur de tester les primitives de base qui lui seront disponibles, et de tester par l'intermédiaire du prototypage des interfaces leurs interactions avec les fonctions de plus haut niveau.

Parmi ces trois types de prototypage, le prototypage fonctionnel est celui qui fait le principal objet de recherche dans le domaine des méthodes de développement de logiciels. Etant donné le coût de la maintenance fonctionnelle, un effort particulier est fourni pour améliorer les méthodes de développement. Non seulement l'introduction du prototypage fonctionnel dans les méthodes de développement a comme effet de réduire ce coût de maintenance, mais il est également un outil de communication essentiel entre utilisateurs et concepteurs dans le but de déterminer efficacement les caractéristiques et les fonctions que devra satisfaire le système final.
Le processus de prototypage du dialogue est celui qui est le plus utilisé, étant donné les exigences croissantes des utilisateurs en matière de dialogue homme-machine.
Par contre, le processus de prototypage des performances est le moins utilisé. Plus les systèmes évoluent, plus les contraintes sur les performances sont importantes, mais les contrôles à effectuer dans le cadre du prototypage des performances sont très difficiles.
Le domaine d'application de ces deux types de prototypage est limité à des aspects particuliers du système, pour lesquels la nécessité de l'application du processus de prototypage ne se fait ressentir que si ces aspects sont des caractéristiques importantes du système à développer. Leur impact sur le processus de développement et sur le système final résultant est de moindre importance que celui du prototypage des fonctions que le système doit réaliser.
La limitation de l'application du système de prototypage DSL-PROTO au prototypage fonctionnel ne constitue donc pas véritablement une restriction du domaine d'application du système.

Si d'une part il est interessant d'analyser le domaine d'application du système de prototypage, d'autre part il est

également important de préciser les classes d'utilisateurs susceptibles d'être interessés par le processus de prototypage ou par le prototype produit.
Le système de prototypage DSL-PROTO s'adresse principalement aux utilisateurs organisationnels et analystes. Ceux-ci ont pour tâche de déterminer les caractéristiques fonctionnelles du système d'information, d'évaluer les différentes solutions alternatives au problème posé et de valider la spécification des besoins définis par les utilisateurs finals.
D'autres utilisateurs, les informaticiens, sont concernés par le produit résultant, parce qu'il constitue une base précise pour la spécification de l'implémentation du système. Il est donc utilisé comme point de départ des étapes suivantes du processus de développement.
Par contre, les utilisateurs exploitant du système final ne sont pas concernés par le prototypage fonctionnel. Leur interaction avec le système et le produit final se limite au dialogue avec la machine. Or le prototypage de l'interface homme-machine n'est pas réalisé par le système DSL-PROTO.

2.4.3 Critiques des objectifs

Nous nous proposons d'analyser successivement les différents objectifs assignés au système de prototypage DSL-PROTO, à savoir la fonction de reproduction du système réel, la fonction de validation expérimentale des spécifications et la fonction d'apprentissage des règles d'utilisation du système.

2.4.3.1 Reproduction du comportement réel

Le premier objectif assigné au système de prototypage est de fournir à l'utilisateur une vue globale du comportement du système d'information par le prototypage des fonctions du système d'application et des flux des messages ainsi que des interactions entre traitements manuels et automatisés. Pour satisfaire cet objectif, le système doit être capable de reproduire le comportement réel du système d'information final sur base des spécifications des besoins définies par l'utilisateur.
La version actuelle du système ne permet pas la reproduction du comportement réel du système d'information. En effet, DSL-PROTO permet l'évaluation expérimentale des fonctions du

système par création et exécution d'une maquette du système dans un environnement simplifié. Les caractéristiques détaillées de cette exécution sont présentées au paragraphe 2.3.4.. Une telle exécution dans un tel environnement ne peut que donner une vue parfaite , trop optimiste, du comportement du système. Les caractéristiques de l'exécution d'un système à échelle réduite sont des contraintes importantes qui ne permettent en aucun cas la reproduction du comportement réel.

Les contraintes caractéristiques de l'environnement de l'exécution sont les suivantes:

- la gestion et le stockage d'un volume réduit de données en mémoire centrale. Il est évident que dans ce cas, les performances à l'exécution sont les meilleures possibles, mais ne correspondent pas aux performances réelles.

- l'exécution du système dans un environnement mono-utilisateurs. L'exécution ne permet donc pas de tester le comportement du système dans un environnement multi-utilisateurs. Or, plusieurs utilisateurs exécutant des tâches qui manipulent les mêmes données influencent le comportement et les performances du système.

- pas de prise en charge des problèmes de fiabilité et de sécurité pendant l'exécution, tels que les problèmes de points de reprise et de copies de sécurité. Les problèmes de cohérence des données lors de l'exécution dans un environnement multi-utilisateurs sont des aspects qui influencent le comportement du système et dont il faut tenir compte.

Ces caractéristiques contribuent à donner aux utilisateurs une fausse idée du comportement réel du système. En fait, une telle exécution ne permettra jamais au système de prototypage de fournir aux utilisateurs une vue du comportement réel du système d'application. Le comportement sera celui de l'exécution d'une maquette du système dans un environnement simplifié et ne correspondra pas à un environnement réel d'exécution de sytème.

En conclusion, nous pouvons constater que l'objectif de reproduction du comportement réel du système d'information n'est pas atteint. Toutefois, cet objectif semble être réalisable et il est évident que la suppression de ces contraintes serait un pas important dans cette direction.

2.4.3.2 Validation expérimentale

Cette fonction doit permettre aux utilisateurs de tester expérimentalement les spécifications du système d'information pendant son développement. L'accent est mis sur la validation et l'amélioration des spécifications, et plus particulièrement des règles de traitement de l'information, par une expérimentation directe de la solution proposée.

L'utilisateur doit pouvoir exécuter le prototype du système et vérifier que l'exécution des spécifications fournit les résultats attendus. En cours d'exécution, et plus particulièrement avant et après l'exécution de chaque procédure de prototypage, l'utilisateur doit avoir la possibilité de visualiser l'état de la mémoire du système, et de contrôler que l'exécution a produit les résultats attendus sur les données. Si l'utilisateur n'est pas satisfait, il doit pouvoir facilement apporter des modifications aux spécifications et rapidement les retester.

Nous pouvons constater en analysant la présentation du test d'effectivité au paragraphe 2.3.5. que l'objectif de validation expérimentale est entièrement atteint par le système de prototypage DSL-PROTO. La possibilité donnée à l'utilisateur de fixer le niveau de trace qu'il désire lui permet de vérifier les résultats des traitements effectués sur les données non seulement avant et après l'exécution du prototype, mais aussi pendant son exécution, pour chaque processus exécuté. Ainsi l'utilisateur peut comparer à n'importe quel niveau de l'exécution, les résultats obtenus et ceux attendus et émettre une opinion et une conclusion sur l'effectivité du système.

2.4.3.3 Apprentissage des règles d'utilisation

La fonction d'apprentissage des règles d'utilisation du système permet aux utilisateurs d'exécuter plusieurs fois le même prototype du système d'information dans des conditions à chaque fois identiques. Cela signifie que chaque exécution se déroule dans un même environnement.
Cette fonction a pour objectif de permettre un apprentissage des règles d'utilisation du système d'information par la répétition de l'exécution d'un prototype. A l'exécution, l'utilisateur peut spécifier, soit que les mises à jour des

données ne soient pas physiquement enregistrées dans la mémoire du système, et en début de réexécution du prototype, la mémoire se trouve toujours dans le même état que lors de l'exécution précédente, soit que les mises à jour soient cumulées dans la mémoire du système, et dans ce cas l'environnement évolue à chaque exécution.

Ce mécanisme va donc permettre d'envisager deux types d'apprentissage:

- un apprentissage des règles d'utilisation par l'exécution répétée du prototype dans des conditions identiques à chaque exécution,

- un apprentissage des règles d'utilisation par l'exécution répétée du prototype dans un environnement évoluant à chaque exécution.

Le processus d'apprentissage des règles est caractérisé par le fait qu'il travaille sur une copie de la mémoire du système.
Pour le premier type d'apprentissage, les mises à jour effectuées sur les données sont enregistrées dans la mémoire du système, mais en début de réexécution du prototype, le système restitue, sur base de la copie, la mémoire dans le même état que lors de l'exécution précédente.
Tandis que pour le second type, les mises à jour effectuees sur les données sont enregistrées dans la mémoire du système, et l'état de cette mémoire en fin d'exécution du prototype correspond à son état initial pour l'exécution suivante. Ainsi l'utilisateur peut expérimenter les règles d'utilisation du système d'information dans un environnement à chaque fois différent, un environnement qui correspondrait à une évolution plus réaliste de la mémoire du système.
L'apprentissage peut donc se faire en expérimentant à chaque exécution les mêmes règles d'utilisation, soit en expérimentant à chaque exécution des règles d'utilisation différentes étant donné que l'environnement de travail est différent.

2.4.4 Critique de la production du prototype

La production du prototype du système d'information passe par l'écriture des spécifications du système et des procédures de prototypage correspondant aux règles de traitements effectués sur les données. Les spécifications relatives aux structures d'information, aux messages, aux entrées-sorties des fonctions et aux enchainements des

traitements sont en partie automatiquement récupérées par le système de prototypage.
Une paranthèse peut être ouverte pour analyser brièvement les règles d'écriture de la dynamique des processus. Nous avons montré au chapitre précédent que cette dynamique pouvait s'exprimer soit par les processus, soit par les messages. L'une ne présente pas plus d'avantage que l'autre, sauf que l'expression de la dynamique par les messages apparait comme une nécessité. En effet, suivant le principe qui spécifie qu'un processus ne recoit de messages que du processus qui l'a déclenché, et sachant que toute transmission d'information d'un processus à l'autre se fait obligatoirement par des messages, l'expression de la dynamique par les messages est inévitable. Il est important de faire remarquer que les deux types d'expression sont obligatoires. En effet, l'expression par les processus est nécessaire dans certains cas. Par exemple dans le cas où un processus génère plusieurs messages de même type,ce n'est pas chaque génération de message qui déclenche le processus suivant, mais celui-ci n'est déclenché que par la terminaison du processus, c'est-à-dire lorsque tous les messages ont été générés.
Pour obtenir une maquette du système, il reste à transformer l'énoncé des règles de traitements en algorithmes, et à définir les conditions de réalisation des points de synchronisation.

L'énoncé des points de synchronisation est très simple. Le langage d'expression est présenté au paragraphe 2.3.5.2.. La simplicité des opérateurs et des opérandes mis à disposition de l'utilisateur limite terriblement la portée de la réalisation de la synchronisation et empêche l'expression de n'importe quelle synchronisation.

Le langage de prototypage PROTO mis à disposition de l'utilisateur ne peut pas être considéré comme un langage de haut niveau. C'est un langage proche d'un langage de programmation qui impose un certain nombre de contraintes lors de l'écriture des procédures de prototypage. Des exemples de contraintes sont présentées au paragraphe 2.3.5.3..
Nous pouvons en outre constater la contrainte importante suivante. Les limites de la prise en charge automatique et de la traduction automatique des spécifications en instructions du programme de prototypage oblige l'utilisateur à écrire dans les procédures de prototypage des instructions correspondantes à des specifications. Cette contrainte est illustrée dans l'exemple suivant de procédure de prototypage emboitée dans une spécification DSL.

* Process inputs, outputs and dynamics

-LE PROTOTYPAGE DANS IDA-

```
DEFINE PROCESS     Verify-Order ;
  RECEIVES         Order-Voucher ;
  REFERENCES       Client-Identify INTO Client ;
  GENERATES        Accepted-Order-Voucher IF Client-Exists ;
  TRIGGERED BY GENERATION OF Order-Voucher ;
  ON TERMINATION TRIGGERS Client-Creation IF NOT Client-Exists ;
  ON TERMINATION TRIGGERS Order-Registration IF Client-Exists ;

   * Process Procedure

PROTOTYPING-PROCEDURE ;

  PROCEDURE Verify-Order

    DEFINE Index FORMAT IS INT (2)

    DEFINE I FORMAT IS INT (2)

    RECEIVES Order-Voucher

      LET Index := CARDINALITY (Order-Line OF Order-Voucher)

      FOR EACH Client
        WHERE Client-Identify OF Client =
              Client-Identify OF Order-Voucher

        LET Client-Exists := T

        OPENGENERATES Accepted-Order-Voucher

          LET Client-Name OF Accepted-Order-Voucher :=
              Client-Name OF Order-Voucher

          LET Client-Identify OF Accepted-Order-Voucher :=
              Client-Identify OF Order-Voucher

          LET I := 0

          WHILE I <= index
            LET I := I + 1
            LET Item-Number OF Order-Line (I)
                OF Accepted-order-Voucher :=
                Item-Number OF Order-Line (I) OF Order-Voucher

            LET Order-Quantity OF Order-Line (I)
                OF Accepted-Order-Voucher :=
                Order-Quantity OF order-Line (I) OF Order-Voucher
          ENDWHILE

        GENERATES Accepted-order-Voucher

        WHEN NONE

          LET Client-Exists := F
```

```
        ENDFOR Client

    ENDRECEIVES Order-Voucher

  ENDPROC Verify-Order
```

Par l'analyse de cet exemple, nous pouvons aisément constater que l'expression de mêmes faits , à savoir la réception et la génération d'un message, apparaissent deux fois, une fois dans les spécifications proprement dites, et une fois dans la procédure. Il faudrait que le système de prototypage puisse déduire de manière automatique des instructions du programme de prototypage à partir des spécifications.
Une deuxième contrainte est l'obligation de définir et d'utiliser des variables pour référencer des types d'objets qui sont définis et connus au niveau des spécifications.
Dans l'exemple précité, le type d'objet ORDER-LINE est défini dans les spécifications, mais il n'est pas directement référencé dans la procédure de prototypage. Les variables I et INDEX ont dû être définies et utilisées pour le référencer. L'instruction WHILE I<=INDEX devrait pouvoir être remplacée par l'instruction FOR EACH ORDER-LINE OF ORDER-VOUCHER.

Une amélioration du langage est indispensable pour faciliter l'écriture des procédures de prototypage et pour améliorer la qualité et l'efficacité des algorithmes et des procédures qui seront générées et utilisées lors de l'exécution du prototype.

2.4.5 Critique de l'évaluation du prototype

Une première remarque concerne le fait qu'une fois que l'état de la mémoire a été initialisé et q'une première exécution s'est déroulée, dans le but de maintenir la cohérence des données, l'utilisateur n'a plus accès à la mémoire du système. Seuls les programmes de prototypage peuvent y accéder pour y effectuer des mises à jour.
Ce choix permet de garantir à chaque exécution de prototype la cohérence des données de la mémoire du système.

Une deuxième remarque concerne l'intervention de l'utilisateur pendant l'exécution du prototype.
Cette intervention est obligatoire lorsqu'un processus doit recevoir un message qui n'a pas encore été généré. L'utilisateur doit alors entrer lui-même les données relatives à ce message.

D'autre part, le concepteur peut avoir explicitement programmé l'intervention de l'utilisateur par une instruction de saisie de données. Il est important de faire remarquer que toute saisie de données se fait par l'intermédiaire de messages, qui doivent être prédéfinis par le concepteur lors de l'écriture des spécifications.
Le prototypage des processus interactifs pourraient être possible. Le dialogue devrait se faire par des messages, et le concepteur aura dû décrire autant de messages qu'il y a de dialogue question-réponse. Ce mécanisme de prise en charge du prototypage des processus interactifs est assez fastidieux, car il nécessite l'introduction de messages fictifs pour prototyper le dialogue. De plus, il pose un problème au niveau de la gestion des écrans. Cette simulation du dialogue par échange de messages ne permet pas une vraie gestion des écrans. Une proposition de solution est brièvement présentée au chapitre suivant.

2.4.6 Critiques de l'utilisation future du prototype

Produire et exécuter un prototype d'un système d'information comprend la spécification des besoins des utilisateurs, la spécification des traitements, des informations manipulées et des ressources nécessaires, et l'écriture des procédures qui correspondent aux traitements effectués sur les données. Ces procédures sont écrites en langage PROTO et sont traduites en programmes exécutables.

Actuellement, l'utilisation du prototype après son évaluation et son expérimentation se limite à peu de chose. D'une part, le prototype sert d'outil d'apprentissage et d'expérimentation des spécifications du système d'information et est jeté après utilisation. D'autre part, les algorithmes produits servent de base précise aux spécifications de l'implémentation du système final.
Le code généré pour l'exécution du prototype est donc considéré comme du code-à-jeter. Cela signifie qu'il n'est pas utilisé dans la production du système final.

Il est vrai que le travail fourni pour produire le prototype et l'expérimenter, valider et améliorer les spécifications est utile et nécessaire et que les résultats obtenus sont positifs. Mais il serait interessant d'optimiser l'utilisation des programmes dérivés des procédures de prototypage, et de ne pas limiter l'utilisation des programmes générés aux spécifications pour l'implémentation future. L'exécution du prototype a montré

que ces programmes ont une certaine efficacité. Il est évident qu'il est impossible de les utiliser tels quel, mais des améliorations au langage, aux primitives de manipulation des données et aux structures algorithmiques fournies pourraient permettre d'envisager une traduction automatique de tout ou d'une partie de ces programmes en composants du code final du système en développement.

2.4.7 Conclusion

Il est important de remarquer que la plupart des critiques et des objections émises ne sont pas sans solution. Nous nous proposons dans le chapitre suivant de présenter des améliorations au système de prototypage DSL-PROTO dans le but de réaliser les objectifs assignés, d'obtenir des programmes de prototypage de manière plus automatique en améliorant le langage disponible, et de permettre une utilisation du code généré pour l'exécution du prototype dans le système final.

2.5 BIBLIOGRAPHIE

- [BERTINCHAMPS] J.P. Bertinchamps, M.H. Gatellier :
  "Système de pilotage pour la conception d'un schéma conceptuel d'information"
  Mémoire de 3ième Licence et Maitrise en Informatique, Facultés Universitaires Notre Dame de la Paix, Namur.

- [BLIN] M. Blin :
  " IDA "
  Génie Logiciel numéro 3 : "Maquettage et prototypage : outils et techniques ".

- [BOD85J] F. Bodart, Y. Pigneur, A.M. Hennebert, J.M. Leheureux :
  " The experimentation of information system requirements by simulation and prototyping in the IDA project "
  ISDOS - IDA - PRISE Meeting, Paris January 1985.

- [BOD85D] F. Bodart, Y. Pigneur, A.M. Hennebert, J.M. Leheureux :
  " Computer-aided specification, evaluation and monitoring of information systems "
  The sixth International Conference on Information Systems, Indianapolis December 1985.

- [PIGNEUR] Y. Pigneur :
  "Modélisation et évaluation du comportement dynamique d'un système d'information"
  Thèse de doctorat, Namur, 1984.

- [RAPP] Rapport de synthèse sur le langage PROTO.

- [MANREF] Manuel de référence DSL-PROTO.

CHAPITRE 3 : EXTENSIONS DE DSL-PROTO

3.1 Introduction

A la suite de l'évaluation de l'outil de prototypage DSL-PROTO présentée au chapitre précédent, nous nous proposons de présenter quelques améliorations possibles du système en réponse aux critiques émises et comme solutions aux problèmes soulevés.

Dans un premier point,nous présenterons une modification du système de gestion des données dans le but de satisfaire les objectifs du système de prototypage.

En second lieu, nous proposerons brièvement une extension du langage de prototypage. Une modification des primitives et des structures algorithmiques disponibles faciliterait à l'utilisateur non expérimenté l'écriture des procédures de prototypage et une amélioration des techniques de traduction automatique minimiserait les divergences entre le système spécifié et le prototype produit.

En troisième lieu, nous présenterons une extension possible du système d'evaluation du prototype, qui permettrait le prototypage effectif des processus interactifs.

Dans un dernier point, nous envisagerons une possibilité de récupération du code généré pour l'exécution du prototype dans le but d'une utilisation comme composant du système final. Cette évolution permettrait au système de prototypage de générer directement à partir des spécifications du système et des programmes dérivés des procédures de prototypage le code final du système d'application, c'est-à-dire de faire du système de prototypage un générateur d'application.

3.2 Réalisation des objectifs

3.2.1 Reproduction du comportement réel

Etant donné l'environnement d'exécution proposé, le système DSL-PROTO est plus un outil de maquettage qu'un outil de prototypage proprement dit [BLIN]. En effet, DSL-PROTO permet aux utilisateurs de vérifier et de valider expérimentalement les spécifications du système d'information en cours de développement par réalisation et exécution d'une maquette du système et non d'un prototype, c'est-à-dire d'une version exécutable du système d'information futur. Dans le but de satisfaire l'objectif de reproduction du comportement réel du système d'information, ou du moins de s'en rapprocher, nous proposons que le principe de réalisation et d'exécution d'une maquette du système soit remplacé par un principe de génération automatique et d'exécution d'un prototype du système d'information. (a)

→ partiellement

Les contraintes imposées par l'exécution d'une maquette dans un environnement tel que celui décrit au chapitre précédent peuvent être levées par la transformation de la gestion des données sous forme de tables en mémoire centrale, en une gestion des données à l'aide d'un système de gestion de base de données.
Nous ne nous étendrons pas plus sur la spécification et la réalisation de cette extension étant donné qu'elle fait l'objet d'une étude approfondie exposée dans la deuxième partie de ce mémoire.

(a) Nous utiliserons par le suite le terme 'maquette' lorsque nous référencerons la maquette générée et exécutée dans le système actuel, et nous utiliserons le terme 'prototype' lorsque nous référencerons le système prototypé tel que décrit dans la littérature et tel que nous l'envisageons comme extension du système de prototypage actuel.

3.2.2 Langage de prototypage

L'analyse au chapitre précédent a montré l'importance des restrictions et des contraintes imposées aux utilisateurs par le langage utilisé pour l'écriture des procédures de prototypage. L'écriture de ces procédures devrait être facilitée par la mise à disposition de l'utilisateur d'un langage de haut niveau et par l'utilisation de techniques et de mécanismes permettant une dérivation automatique de certaines instructions à partir des spécifications écrites en DSL.

Nous nous proposons d'exposer brièvement les modifications ou évolutions possibles des différents aspects de l'écriture du prototype, à savoir la prise en charge des phrases DSL, l'écriture des points de synchronisation et l'écriture des procédures de prototypage.

Possibilités d'évolution des phrases DSL utilisables dans un programme maquette:
Sur base de la description statique des processus et des types d'objets, un maximum d'éléments du programme de prototypage devrait automatiquement être produits. La réception et la génération des messages, la description en DSL des manipulations effectuées sur les types d'objets par les processus devraient être automatiquement traduites en instructions du programme de prototypage, sans imposer à l'utilisateur la réécriture de ces descriptions dans la procédure de prototypage.

Possibilité d'évolution de l'écriture des points de synchronisation:
Les conditions de synchronisation peuvent être exprimées à l'aide des opérateurs unaires et binaires suivants, AND, OR, SAME-ORIGIN, COUNT et GROUP-ALL et des opérandes suivantes, terminaison d'un processus, activation d'un processus, génération d'un message ou réalisation d'un point de synchronisation. Les conditions de synchronisation ne peuvent donc pas porter sur des valeurs spécifiques de types d'objets,ce qui limite la portée de la réalisation et réduit la possibilité d'exprimer n'importe quelle synchronisation. L'introduction de la notion de procédure pour exprimer la condition de réalisation des points de synchronisation permettrait d'exprimer n'importe quelle synchronisation. Un point de synchronisation serait réalisé par exécution de la procédure dans laquelle seraient définis la condition de synchronisation, l'événement causé par la réalisation du point de synchronisation, les événements coordonés par cette condition de synchronisation et les modalités de contribution d'un événement à une synchronisation.

Possibilité d'évolution de l'écriture des procédures de prototypage:
La production d'un prototype passe par l'écriture des programmes qui réalisent les différentes fonctions du système d'information. PROTO est un langage d'expression d'algorithme de haut niveau dans lequel sont intégrées des primitives de manipulation des structures de données. Le langage PROTO en tant que langage de programmation met à disposition de l'utilisateur des primitives de manipulation de données et des structures algorithmiques qui permettent d'écrire les procédures de prototypage associées aux processus du prototype.
L'utilisation d'un langage adéquat, d'utilisation simple facilite l'écriture de ces programmes.
Un tel langage serait plus proche d'un langage de requêtes que d'un simple langage de programmation.
Un langage de programmation fournit aux utilisateurs des structures algorithmiques et des primitives de manipulation des données pour l'écriture des programmes.
Un langage de requêtes par contre permet, pour accéder aux données et pour les manipuler, de décrire des ensemble de données en n'en définissant que les propriétés, c'est à dire sans faire référence aux techniques d'accès qui sont utilisées pour atteindre les données de la base. Cette désignation d'ensemble de données prend la forme de requêtes.
Les procédures de prototypage pourraient se présenter sous la forme suivante:

SI condition (requête sur message et/ou requête sur base de données)
ALORS exécution primitive de manipulation ou de définition de données et/ou appel à une procédure

3.3 Evaluation du prototype

Une modification importante à considérer lors du test d'effectivité serait la possibilité de prototyper efficacement les processus interactifs. Une proposition de réalisation de cet objectif, qui nécessiterait une recherche et une étude approffondie, serait de combiner un outil de prototypage fonctionnel et un outil de prototypage du dialogue. Le premier permettant l'exécution du processus interactif et la saisie des données, le second permettant le prototypage du dialogue et des écrans.

3.4 Utilisation future

L'utilisation du prototype généré après son évaluation peut dépendre de plusieurs facteurs, tels que par exemple les résultats obtenus par l'expérimentation et l'évaluation, ou l'environnement de travail et les outils disponibles. En général, le prototype sert d'outil d'apprentissage et d'expérimentation des spécifications du système d'information et est jeté après utilisation.

L'évaluation au chapitre précédent a montré qu'il serait interessant de pouvoir utiliser le code généré pour l'exécution du prototype dans la version finale du système. Réutiliser le prototype de cette manière n'est pas toujours possible.
Il se peut que le prototype soit écrit dans un langage de très haut niveau qui fournit des facilités d'expression et de modification rapide, mais qui ne fournit pas de concepts utilisables dans des programmes plus importants.
Dans certains cas, il serait possible de modifier quelque peu les programmes de prototypage pour pouvoir les introduire dans le système final. Cette transformation pourrait se faire de manière automatique si les outils nécessaires sont disponibles. Cela signifie que les programmes ou certaines procédures du prototype pourraient être automatiquement transformées en procédures du système final.

Dans le cadre du système de prototypage DSL-PROTO, récupérer et utiliser le code généré pour le prototypage dans le système final consiste à assigner une nouvelle fonction au système, la fonction de génération de code final, à faire du système de prototypage un système de génération d'applications.
Actuellement l'exécution du prototype est contrôlée par un moniteur, qui est un processeur d'événements. Ce sont les événements survenant dans le système qui guide l' exécution du prototype. Celle-ci se déroule de la facon suivante.
Le moniteur constate la survenance d'un événement. Cet événement peut correspondre à la génération d'un message qui déclenche l'exécution d'un processus. Le processus qui doit être exécuté est mis par le moniteur dans un file d'attente, il est exécuté par appel à la procédure qui les réalise. A chaque processus sont associés ses conditions de déclenchement et les messages auxquels il peut accéder. Ceux-ci sont les messages qui ont été générés par le processus exécuté précédemment.
Evoluer vers un système de génération d'applications par transformation du code-à-jeter en code final correspond à la transposition des programmes de prototypage, générés à partir des spécifications et des procédures de prototypage, en procédures du programme final de l'application.
Cette évolution du système d'exécution de prototype vers un système de génération de code final nécessite l'introduction d'un mécanisme de transformation automatique ou partiellement automatique du code-à-jeter en code final. Cela implique la prise en charge par le système de prototypage de caractéristiques du système final qui n'étaient pas considérées ou qui étaient simplifiées au niveau du prototype.

Une première modification à apporter au système actuel est la transformation du système de gestion des données en système de gestion de base de données.
Cette modification nécessite l'introduction dans le système d'un module de gestion des accès, interface entre le système de gestion de base de données et le système d'application. Le systeme de gestion des accès aux données, caractérisé par un ensemble de primitives d'accès, peut être utiliser tel quel dans le système final. Chaque accès aux données correspond à l'exécution d'une primitive du système de gestion.
Cette modification donne également la possibilité au système de gérer un volume important de données et de disposer d'une mémoire extensible à souhait.

Un deuxième problème à résoudre est la réalisation des points de synchronisation. Dans le but de pouvoir exprimer n'importe quelle synchronisation, la notion de procédure de réalisation devrait être introduite. Cette procédure

correspondrait à un processus endormi qui serait réveillé chaque fois que la survenance d'un événement contribuant à la réalisation est détecté.
Deux types de problèmes sont à considérer. Le premier apparait quand plusieurs points de synchronisation sont en attente de réalisation, il faut associer les contributions avec les synchronisation correspondantes. Le deuxième type de problème est la détection de la réalisation du point de synchronisation.
Deux types de modification doivent être prise en compte. La première est la modification des règles d'écriture des conditions de réalisation des points de synchronisation par l'introduction de la notion de procédure. La deuxième est la prise en charge de cette procédure par le moniteur.

Un troisième aspect à considérer est la prise en charge du parallélisme qui peut apparaître dans les spécifications de la dynamique des processus. Le parallélisme correspond soit au mécanisme d'enchainement éclaté, où la survenance d'un événement provoque le déclenchement simultané de plusieurs processus, soit au mécanisme d'enchainement multiple, où la survenance d'un événement provoque le déclenchement simultané de plusieurs processus de même type, c'est-à-dire provoque plusieurs exécutions du même processus [BODART].
L'exécution en parallèle n'est actuellement pas possible. Il est remplacé par un mécanisme d'exécution séquentielle. Sous le contrôle du moniteur, les processus sont déclenchés en parallèle, mais ils sont exécutés en série, suivant l'ordre de définition dans les spécifications.
Dans le contexte de génération de code final, la modalité de prise en charge ne peut pas être laissée à un mécanisme automatique, mais elle doit être prise en accord avec l'utilisateur exploitant.
Deux options peuvent être envisagées:

- avertir l'utilisateur exploitant de la prise en charge du parallélisme de manière séquentielle en précisant les règles de prise en charge;

- les différentes séquences possibles d'exécution des processus sont exposées à l'utilisateur exploitant sous forme de menus, et le choix lui est laissé de déterminer la séquence des processus à exécuter.

3.5 Conclusion

A titre de conclusion, nous pouvons faire remarquer que cette brève présentation des extensions et des évolutions possibles du système de prototypage ne doit être considérée qu'à titre exemplatif, les solutions proposées ne prétendant pas être les seules possibles.

3.6 BIBLIOGRAPHIE

- [RIDDLE] W. E. Riddle :
  "Advancing the state of the art in software system prototyping"
  from "Approaches to prototyping" : Proceedings of the working conference on system prototyping, Namur October 1984.

- [BODART] F. Bodart, Y. Pigneur :
  "Conception assistée des systèmes informatiques:
  Tome 1 : Etude d'opportunité et analyse fonctionelle" Masson 1983

DEUXIEME PARTIE

LES EXTENSIONS DE DSL-PROTO

CHAPITRE 4 : GESTION SOUS UNE BASE DE DONNEES RELATIONNELLE

4.1 Objectifs

L'objectif du système de prototypage DSL-PROTO est de fournir à l'utilisateur un environnement de travail qui lui permette d'exécuter les spécifications du système d'information, soit dans le but de tester un prototype de ce système, soit dans le but de générer du code final qui sera intégré en tout ou en partie dans le sytème final.

4.1.1 Exécution d'un prototype

Alors qu'une maquette est simplement un modèle réduit, une reproduction à échelle réduite, une représentation schématique du système à réaliser, un prototype est un exemplaire du système destiné à expérimenter en fonctionnement ses qualités en vue de son utilisation définitive.

La gestion du système de prototypage sous une base de données va permettre de remplacer le principe de réalisation d'une maquette par l'expérimentation d'un prototype du système, et ce en supprimant les contraintes imposées par la réalisation d'une maquette (cfr paragraphe 2.4.4.1.).

L'exécution d'un prototype sous une base de données relationnelle va en effet permettre au système de gérer un volume de données beaucoup plus important puisque celles-ci ne seront plus stockées en mémoire centrale mais dans une base de données extensible à souhait.
De même la structure et l'organisation des données dans une base de données relationnelle va permettre de donner à l'utilisateur une vue plus réaliste des performances d'accès aux données et de

se rapprocher de la réalisation de l'objectif de reproduction du comportement réel du système.

Elle permettra également de considérer l'exécution du prototype dans un environnement multi-utilisateur. La notion de transaction va permettre à chaque utilisateur d'accéder à la base de données pour effectuer ses tâches et va permettre au système de contrôler leurs activités pour éviter des conflits d'accès et des données incohérentes. La gestion de la notion de transaction est discutée au chapitre suivant.

Les problèmes de sécurité et de fiabilité à l'exécution peuvent être pris en considération. En effet, une transaction est une unité de travail qui doit être exécutée entièrement ou ne pas être exécutée du tout. Elle détermine donc des unités d'exécution dont les points de début et de fin constituent des points de reprise. D'une part, en cours d'exécution, les mises à jour ne sont effectivement effectuées qu'aux points de reprise et à la suite de tout incident survenant entre deux points de reprise, un retour est possible au point de reprise précédant, c'est-à-dire au dernier point de reprise où un enregistrement effectif a été réalisé.

4.1.2 Génération de code final

Dans la version actuelle du système de prototypage DSL-PROTO, l'énoncé des règles de traitement est transformé en un algorithme écrit en langage de prototypage. Les algorithmes ainsi produits sont compilés et traduits en code exécutable qui correspond à du 'code-à-jeter'. Cela signifie que le code généré pour l'exécution du prototype n'est pas utilisé dans la version définitive. Le procédé de prototypage peut être considéré comme un processus d'apprentissage et de test du système d'information.

Dans le dernier chapitre de la première partie de ce mémoire, nous avons présenté le problème de l'évolution du système de prototypage DSL-PROTO vers un système de génération de code final. Dans ce cas, les algorithmes générés sont utilisés entièrement ou partiellement comme composants du système final, le code généré correspond alors à du code final.

Nous avons montré que cette évolution nécessite un certain nombre de modifications.
La première consiste à transformer le système de gestion des données en mémoire centrale par un système de gestion de base de données dans le but de maximiser la transformation automatique du

code-à-jeter en code final.
Les deux autres modifications importantes à prendre en considération, qui sont présentées au chapitre précédent, sont la prise en charge de la réalisation des points de synchronisation par exécution d'une procédure de synchronisation, et la prise en charge du parallélisme qui peut apparaitre dans les spécifications de la dynamique des processus.

Un premier pas vers la réalisation de cet objectif sera effectuée par la structuration de la mémoire du système en base de données et par l'introduction dans l'architecture du système d'un module de gestion des accès aux données.

4.2 Définitions

4.2.1 Modification de la structure de la mémoire

La version actuelle du système de prototypage DSL-PROTO travaille avec une mémoire structurée en listes chainées. A chaque type d'entité, d'association et de message pour lesquelles des occurrences sont enregistrées correspond une liste chainée d'occurrences gérée et stockée en mémoire centrale.

Cette structure de listes chainées gérée en mémoire centrale est remplacée par une base de données relationnelle dont la structure correspond à la structure des données définie dans les tables de définition.

4.2.1.1 Justification du choix de la structure

Les types de données manipulés par les processus du système d'information sont exprimés en langage DSL suivant le modèle entité/association. Ces types de données sont ensuite exprimés suivant le modèle relationnel et leur définition est enregistrée dans les tables de description de la structure de données. Les occurrences de ces types de données sont gérées et stockées en mémoire centrale.

Le choix d'une structure de base de données relationnelle est justifié par le fait que le modèle relationnel est plus proche du modèle entité/association que n'importe quel autre modèle de base de données. L'idéal serait une base de données dont celles-ci sont organisées suivant le modèle entité/association, qui permettrait un 'mapping' immédiat entre la spécification des données et la structure de la base de données. Aucune base de données n'utilisant le modèle entité/association comme modèle de structure de données, le choix s'est porté sur le modèle relationnel qui en est le plus proche et qui caractérise la structure et l'organisation des données des bases de données relationnelles.

4.2.2 Définition du module d'accès

La gestion du système de prototypage sous une base de données relationnelle est réalisée par un module d'accès qui forme un interface entre le programme d'application DSL-PROTO et la base de données relationnelle.

Un module de gestion de données est une entité exécutable à laquelle le programmeur peut faire appel lorsqu'il désire effectuer une opération sur une base de données [HAINAUT]. Un tel module constitue un SGBD virtuel, défini par la structure des données qu'il gère, par les primitives qu'il offre pour gérer ces données et pour y accéder, et par les règles d'enchainement de ces primitives. Toutes les instructions d'accès à la base de données sont donc isolées du programme d'application et en cas de changement de la structure de la mémoire du système, il suffit de modifier les appels dans le module d'accès sans qu'aucun changement ne doive être introduit dans le programme d'application. Il y a indépendance totale des programmes par rapport à la base de données.

Le module d'accès est divisé en deux groupes de sous-modules. Le premier groupe comporte les modules qui permettent la création de la structure de la base de données, ainsi que les modules qui permettront si nécessaire de modifier cette structure.
Le deuxième groupe contient les modules de gestion des accès et des mises à jour de la base de données.

4.2.3 Définition du contenu de la base de données

La base de données relationnelle permet l'enregistrement des occurrences des types d'entités et d'associations qui sont manipulés par les procédures lors de l'exécution du prototype. Sa structure correspond à la structure des types d'objets définis dans les tables de description. Une correspondance biunivoque peut être établie entre les types d'objets de la structure de données et la structure propre à la base de données relationnelle. Cette correspondance est une correspondance directe entre les structures sans recherche de performance ou de normalisation, et ce dans le but d'obtenir une structure de base

de données qui soit la plus proche possible de la structure des types de données définie dans les tables de description.

Cette correspondance peut être obtenue par application des règles de transformation suivantes:

- à tout type d'entité correspond une table;
- à tout type d'association avec attribut(s) correspond une table;
- à tout type d'association sans attribut correspond une vue; Cette correspondance se justifie par le fait qu'une vue est une table virtuelle définie sur un ou plusieurs champs de deux ou plusieurs tables dont seule la définition est enregistrée dans la base de données. Elle exprime donc, comme le fait une association sans attribut, un lien explicite entre deux ou plusieurs tables sans avoir aucune valeur propre.
- à tout attribut d'un type d'entité ou d'association correspond un champ dans la table correspondant au type d'entité ou d'association,
  - à tout attribut simple correspond un champ;
  - à tout attribut répétitif correspond autant de champs qu'il y a de répétitions; (il n'est admis qu'un nombre fini de répétitions)
  - à tout attribut élémentaire correspond un champ;
  - à tout attribut décomposable correspond un champ par composant;
  - à tout attribut facultatif correspond un nombre de champs suivant le type de l'attribut avec une valeur par défaut pour chaque champ.

Gestion des contraintes d'intégrité.

Les seules contraintes d'intégrité qui sont prises en compte dans les tables de définition de la structure de données sont les contraintes de connectivité et les identifiants.

Gestion des connectivités:

Celles-ci posent une contrainte sur le nombre d'occurrences d'un type d'objet en fonction du nombre d'occurrences d'un autre type d'objet. Le système de gestion de base de données ne permet pas d'exprimer ce type de contrainte. Il est donc nécessaire

d'en reporter la gestion à un niveau supérieur, comme cela est réalisé dans la version actuelle.

Gestion des identifiants:

Identifiants d'un type d'entité:

Les attributs constituant l'identifiant peuvent appartenir au type d'entité à identifier et/ou aux types d'entités qui lui sont reliées via des types d'association (ce deuxième type d'identifiant n'est pas autorisé en DSL-PROTO). Le ou les identifiants sont définis comme index sur la table correspondant au type d'entité qu'il(s) identifie(nt).

Identifiants d'un type d'association:

Type d'association avec attribut: dans ce cas, il faut considérer comme identifiant du type d'association les identifiants implicites, c'est-à-dire les identifiants des types d'entités sur lesquelles le type d'association est défini, ainsi que les identifiants explicites, c'est à dire les identifiants du type d'association lui-même. Cet identifiant global est défini comme index sur la table correspondant au type d'association qu'il identifie.

Type d'association sans attribut: dans ce cas, il n'y a pas d'identifiant explicite et les identifiants implicites sont ceux qui correspondent aux attributs identifiants des types d'entités sur lesquels le type d'association est défini et qui correspondent aux champs sur lesquels est défini la vue qui correspond au type d'association.

peut-on définir un index sur une vue ?

4.3 Spécifications du module de gestion

4.3.1 Introduction

Le module d'accès à la base de données relationnelle se compose de deux parties distinctes:

- le module de création et de modification de la structure de la base de données relationnelle;
- le module de gestion des interrogations et de mise à jour de la base de données relationnelle.

Ces modules sont composés d'un certain nombre de fonctions qui seront spécifiées par la suite. Leur processus d'activation est décrit au paragraphe suivant.

4.3.2 Activation des modules

4.3.2.1 Activation du module de création

Après la création initiale des tables de définition de la structure de données, le module de création et de structuration de la base de données relationnelle est activé. Par la suite, chaque modification dans une des tables de définition par ajout d'un ou plusieurs types d'objets entraine une activation du module de modification de la structure de la base de données relationnelle pour une modification éventuelle d'une table ou d'une vue par ajout d'un ou plusieurs champs supplémentaires, ou pour l'ajout d'une nouvelle table ou d'une nouvelle vue.

4.3.2.2 Activation du module de mise à jour

Ce module peut être activé à deux moments différents. Il est activé lors de l'enregistrement des occurrences initiales des types d'entités et d'associations avant l'exécution du prototype. D'autre part, il est activé pendant l'exécution du prototype lorsque celle-ci requiert un accès à la base de données, soit pour un ajout, une suppression ou une modification d'une occurrence d'un type d'objet, soit pour l'interrogation de la base de données.

4.3.3 Spécifications du module de création

Le module de création et de modification de la structure de la base de données relationnelle est composé d'un ensemble de fonctions d'interface:

- fonction d'initialisation de la structure,
- fonction de création d'une table,
- fonction de création d'une vue,
- fonction de création d'un index,
- fonction de création de champs,
- fonction de modification d'une table par ajout de champs,
- fonction de mofification d'une vue par ajout de champs.

4.3.3.1 Fonction d'initialisation de la structure

données en entrée:

table de description } tables actuelles MARDSC
table des contenus } MARDWN

donnée en sortie:

code résultat
- 0: tout est OK
- 1: erreur de définition de la BD
- 2: conflit d'accès
- 3: erreur de définition de la structure

fonction:

pour les types d'objets entités et

associations de la table de description, la fonction crée dans la base de donnés la structure correspondante aux définitions.

pré-condition:

les tables de définition utilisées reflètent la spécification des différents types d'objets.

post-condition:

s'il n'y a pas d'erreur, la structure de la base de données correspond aux définitions des types d'objets contenues dans les tables de définition après application des règles de transformation.

4.3.3.2 Fonction de création d'une table

donnée en entrée:

numéro de ligne du type d'objet dans la table de description

donnée en sortie:

code résultat

- 0: tout est OK
- 1: conflit d'accès
- 2: erreur de définition de la table

fonction:

sur base de la description du type d'objet et de son contenu (ses attributs), la fonction crée une table et les champs qui y sont associés.

pré-condition:

le numéro de ligne correspond à la description du type d'objet dans la table de description.

post-condition:

s'il n'y a pas d'erreur, la table et les champs correspondant au type d'entité et à ses attributs est définie dans la base de données.

4.3.3.3 Fonction de création d'une vue

donnée en entrée:

numéro de ligne dans la table de description correspondant au type d'association

donnée en sortie:

code résultat

- 0: tout est OK
- 1: conflit d'accès
- 2: erreur de définition

fonction:

sur base des descriptions du type d'association et des types d'entités qui y sont reliées, la fonction crée une vue dans la base de données.

pré-condition:

le numéro de ligne dans la table de description correspond à la description d'un type d'association sans attribut.

post-condition:

s'il n'y a pas d'erreur, la vue définie correspond à la définition d'un type d'association sans attribut.

4.3.3.4 Fonction de création d'un index

donnée en entrée:

numéro de ligne dans table de description du type d'objet correspondant à la table pour laquelle un index est créé

Ne faut-il pas donner le no de ligne de l'identifiant que l'on crée ?

donnée en sortie:

code résultat

- 0: tout est OK
- 1: conflit d'accès
- 2: erreur de définition

fonction:

la fonction crée un index sur la table

correspondant au type d'objet défini dans la table de description.

pré-condition:

le numéro de ligne de la table de description correspond à un type d'entité ou d'association avec attribut(s) pour lequel un index est créé.

post-condition:

s'il n'y a pas d'erreur, un index est créé sur la table correspondant au type d'objet défini à la ligne donnée.

4.3.3.5 Fonction de création d'un champ

donnée en entrée:

numéro de ligne dans la table de description de l'attribut pour lequel on crée un champ

si pas quelle entité ou associat ?

donnée en sortie:

code résultat

0: tout est OK

1: conflit d'accès

2: erreur de définition

fonction:

la fonction crée un champ correspondant à l'attribut défini dans la table de description suivant les règles de transformations prédéfinies.

pré-condition:

le numéro de ligne correspond à la définition d'un attribut dans la table de description.

post-condition:

s'il n'y a pas d'erreur, un ou plusieurs champs sont créés dans la base de données.

4.3.3.6 Fonction de modification d'une table

donnée en entrée:

numéro de ligne dans la table de description du type d'objet dont la définition doit être modifiée.

donnée en sortie:

code résultat
- 0: tout est OK
- 1: conflit d'accès
- 2: erreur de redéfinition

fonction:

sur base de la description du type d'objet, de son nouveau contenu, et de la table y correspondant, la fonction ajoute les nouveaux champs à la définition de la table.

pré-condition:

le numéro de ligne correspond à un type d'entité ou d'association dont la définition est modifiée et dont la table existe.

post-condition:

s'il n'y a pas d'erreur la définition de la table est modifiée en fonction des nouveaux champs à ajouter.

4.3.3.7 Fonction de modification d'une vue

donnée en entrée:

numéro de ligne du type d'objet dans la table de description pour lequel la définition de la vue correspondante doit être modifiée.

donnée en sortie:

code résultat
- 0: tout est OK
- 1: conflit d'accès
- 2: erreur de redéfinition

fonction:

sur base des descriptions du type d'association et des types d'entités sur lesquels il est défini, la fonction modifie la définition de la vue dans la base de données.

pré-condition:

le numéro de ligne correspond à la définition du type d'association sans attribut pour lequel la définition de la vue correspondante doit être modifiée.

post-condition:

s'il n'y a pas d'erreur, la définition de la vue est modifiée en y ajoutant les nouveaux champs.

4.3.4 Spécifications du module d'interrogation et de mise à jour

Les différentes fonctions d'interface qui constituent le module sont les suivantes.

- fonction d'enregistrement d'une occurrence d'un type d'entité ou d'association,
- fonction de modification d'une occurrence d'un type d'entité ou d'association,
- fonction de suppression d'une occurrence d'un type d'entité ou d'association,
- fonction de lecture d'une occurrence d'un type d'entité ou d'association.

4.3.4.1 Fonction d'enregistrement

données en entrée:

buffer contenant l'occurrence à enregister
le nom clé du type d'objet.

donnée en sortie:

code résultat

0: tout est OK
1: conflit d'accès
2: erreur d'enregistrement

fonction:

la fonction enregistre l'occurrence du type d'objet référencé par son nom clé et contenue dans le buffer.

pré-condition:

l'occurrence à enregistrer est syntaxiquement correcte.

post-condition:

l'occurrence est enregistrée dans la table correspondant au type d'objet désigné par son nom clé.

4.3.4.2 Fonction de modification

données en entrée:

buffer contenant les modifications d'une occurrence
nom clé du type d'objet.

donnée en sortie:

code résultat
0: tout est OK
1: conflit d'accès
2: erreur de modification

fonction:

la fonction modifie l'occurrence du type d'objet désigné par son nom clé et dont les nouvelles valeurs sont contenues dans le buffer.

pré-condition:

les valeurs de l'occurrence contenue dans le buffer contiennent au moins les valeurs de la clé d'accès à l'occurrence, et toutes les valeurs sont syntaxiquement correctes.

post-condition:
les nouvelles valeurs de l'ocurrence sont enregistrées dans la table correspondant au type d'objet désigné par son nom clé.

4.3.4.3 Fonction de suppression

données en entrée:
buffer contenant les valeurs de la clé d'accès de l'occurrence à supprimer
nom clé du type d'objet

données en sortie:
code résultat
0: tout est OK
1: conflit d'accès
2: erreur de suppression

fonction:
la fonction supprime l'occurrence dont la clé d'accès est dans le buffer , occurrence du type d'objet désigné par son nom clé.

pré-condition:
la clé d'accès de l'occurrence est syntaxiquement correcte et désigne une occurrence du type d'objet référencé par son nom clé.

post-condition:
l'occurrence dont la clé d'accès est contenue dans le buffer est supprimée de la table du type d'objet désigné par son nom clé.

4.3.4.4 Fonction de lecture

données en entrée:
buffer contenant la clé d'accès à l'ocurrence à lire
nom clé du type d'objet

données en sortie:

buffer contenant l'occurrence lue, code résultat

0: tout est OK
1: conflit d'accès
2: erreur de lecture

fonction:

la fonction lit l'occurrence du type d'objet désigné par son nom clé, occurrence dont la clé d'accès est dans le buffer, et met dans celui-ci l'occurrence lue.

pré-condition:

la clé d'accès de l'occurrence à lire contenue dans le buffer est syntaxiquement correcte.

post-condition:

l'occurrence lue contenue dans le buffer correspond à celle dont la clé d'accès était contenue le buffer et dont le type d'objet est désigné par son nom clé.

4.4 Réalisation de l'interface

4.4.1 Introduction

4.4.1.1 Caractéristiques du SGBD utilisé

Le sytème de gestion de base de données relationnelle utilisé pour la réalisation du module de gestion est VAX Rdb/VMS, système de gestion pour VAX opérant sous un système VMS. Rdb/VMS a tous les avantages d'un système de gestion de base de données, y compris un système garantissant la sécurité et la cohérence des données, ainsi que l'optimisation des accès. En tant que système de gestion de base de données, VAX Rdb/VMS a les caractéristiques suivantes:

- Indépendance et cohérence des données:
  Les définitions des données sont extraites des programmes d'application et stockées avec les données elles-même. Le système réduit l'enregistrement de données redondantes, ce qui permet d'assurer que les mises à jour n'entrainent pas une incohérence dans les données.

- Interaction et environnement multi-utilisateurs:
  Plusieurs utilisateurs peuvent accéder aux données en même temps, le système garantissant que les opérations de chacun d'entre eux sur la base de données n'entrainent aucun résultat incorrect pour les autres.

- Intégrité et sécurité des données:
  Rdb/VMS maintient l'intégrité des données en cas d'erreur de la part de l'utilisateur, en cas de pannes de matériel ou de logiciel, et en cas d'utilisation concurrente de la base de données. De plus, le système de sécurité empêche des utilisateurs non autorisés à accéder aux données.

4.4.1.2 Caractéristiques de la programmation des modules

L'utilisation du système de gestion de base de données relationnelle Rdb/VMS dans un programme d'application peut se faire de deux manières:

- par des précompilateurs: Quand un précompilateur est disponible pour le langage utilisé, les instructions Rdb/VMS peuvent être inclues dans le programme d'application presque sous la même forme qu'elle apparaissent dans le programme interactif RDO (Relational Database Operator). On peut alors référencer des variables du programme d'application dans les instructions Rdb. Celles-ci sont compilées en même temps que le programme qui les contient.

- par appel à l'interface interactif "Callable RDO": Cet outil est une routine externe unique qui accepte une instruction Rdb/VMS comme paramètre. Elle est interprètée et exécutée lors de l'exécution du programme d'application qui la contient.

La programmation des modules du module de gestion des accès utilisera le langage Fortran. Un précompilateur est disponible pour ce langage, mais il ne peut être utilisé car le contenu complet des instructions Rdb n'est connu qu'à l'exécution de ces instructions.

4.4.2 Réalisation du module de création

4.4.2.1 Fonction d'initialisation de la structure

procédé:
parcours de la table de description des types d'objets avec application des règles de correspondance aux types d'entités et d'associations.

procédure:
```
initialisation des fichiers de la base de données
tant que pas fin de table de description
    considérer le type d'objet
    si type de message ou type de groupe ou type d'élément
        alors     passer au type d'objet suivant
```

```
si type d'entité
    alors     créer table
si type d'association
    alors     rechercher contenu
              si attribut
                  alors    créer table
                  sinon    créer vue
fin d'initialisation de la base de données
```

4.4.2.2 Fonction de création d'une table

procédé:

le nom de la table correspond au nom du type d'objet
parcours de la table des contenus avec application des règles de correspondance aux types d'attribut.

procédure:

```
initialisation de la table
rechercher référence contenu
tant que pas fin contenu
    rechercher contenu
    rechercher description contenu
    créer champs
créer index
```

4.4.2.3 Fonction de création d'une vue

procédé:

le nom de la vue correspond au nom du type d'objet,
recherche des types d'entité sur lesquels le type d'association est défini, et recherche de leurs attributs identifiants sur lesquels la vue est définie.

procédure:

```
initialisation de la vue
rechercher référence types d'entités
rechercher référence et description identifiants
définir la vue sur les identifiants
```

4.4.2.4 Fonction de création d'un index

procédé:
    le nom de l'index correspond à la concaténation du nom de la table sur laquelle il est défini et du mot 'idx'.

procédure:
    initialisation de l'index
    recherche référence et description ientifiants
    définition de l'index

4.4.2.5 Fonction de création de champs

procédé:
    détermination du nom du champ:

    si type d'élément
       alors si attribut non répétitif
                alors nom du champ est nom de l'élément
                sinon nom du champ est concaténation du nom de
                      l'élément et de l'indice de répétition
    si type de groupe
       alors si groupe non répétitif
                alors creation de champs pour chaque composante
                sinon le nom de chaque composante est la concaténation
                      du nom de la composante et de l'indice de
                      répétition.

procédure:
    rechercher description attribut
    si type d'élément
        alors   si attribut non répétitif
                    alors   initialiser champ
                             définir champ
                    sinon   pour chaque répétition
                                   initialiser le champ
                                   définir nom du champ
                                   définir le champ
    si type de groupe
        alors   si groupe non répétitif
                    alors   pour chaque composante
                                   créer champ
                    sinon   pour chaque répétition
                                   définir le nom de la composante
                                   pour chaque composante créer champ

4.4.2.6 Fonction de modification d'une table

procédé:
la modification correspond à la création de nouveaux champs et à leur ajout dans la définition de la table.

procédure:
```
rechercher description type d'objet
rechercher référence contenu
tant que pas fin contenu
    rechercher description contenu
    si champ n'existe pas dans définition
        alors    créer champ
                 ajouter champ dans définition de table
```

4.4.2.7 Fonction de modification d'une vue

procédé:
la modification consiste en un ajout de champs dans la définition de la vue.

procédure:
```
rechercher description du type d'objet
rechercher description es types d'entités
rechercher référence et description des identifiants
pour chaque champ des identifiants
    si champ n'existe pas dans définition
        alors    ajouter champ dans description de la vue
```

4.4.3 Réalisation des modules de mise à jour et d'interrogation

4.4.3.1 Fonction d'enregistrement d'une occurrence

procédé:
chaque valeur contenue dans le buffer correspond à la valeur d'un attribut et est enregistrée dans le champ correspondant.

procédure:

```
initialisation de l'instruction d'enregistrement
rechercher description du type d'objet désigné
rechercher référence du contenu
tant que pas fin contenu
    rechercher description contenu
    rechercher nom du champ correspondant
    mise à jour de l'instruction avec nom et valeur du champ
terminer instruction
```

4.4.3.2 Fonction de modification d'une occurrence

procédé:

les valeurs de l'occurrence contenue dans le buffer sont les nouvelles valeurs à enregistrer, si l'espace d'une valeur est à blanc, elle ne doit pas être modifiée.

procédure:

```
initialisation de l'instruction de modification
rechercher description du type d'objet désigné
rechercher clé d'accès à occurrence
ajouter clé d'accès dans instruction
rechercher référence du contenu
tant que pas fin contenu
    rechercher description contenu
    si valeur correspondant existe dans buffer
        alors    rechercher nom du champ correspondant
                 mise à jour de l'instruction avec nom du champ
                                                et nouvelle valeur
terminer instruction
```

4.4.3.3 Fonction de suppression d'une occurrence

procédé:

les valeurs contenues dans le buffer correspondent aux valeurs de la clé d'accès de l'occurrence à supprimer.

procédure:

```
initialiser l'instruction de suppression
rechercher description du type d'objet désigné
rechercher clé d'accès dans buffer
mise à jour de l'instruction avec valeurs de clé d'accès
terminer instruction de suppression
```

4.4.3.4 Fonction de lecture d'une occurrence

procédé:

les valeurs contenues dans le buffer correspondent aux valeurs de la clé d'accès de l'occurrence à lire, après exécution le buffer contiendra les valeurs de l'occurrence lue.

procédure:

```
initialisation de l'instruction de lecture
rechercher description du type d'objet désigné
rechercher clé d'accès dans buffer
mise à jour de l'instruction avec valeurs de clé d'accès
rechercher référence contenu
tant que pas fin contenu
    rechercher description contenu
    rechercher nom du champ correspondant
    mise à jour de l'instruction avec nom du champ
terminer instruction
```

4.5 Impact de la réalisation du module de gestion

La gestion du système de prototypage sous une base de données relationnelle s'est traduite par la restructuration de la mémoire du système en base de données relationnelle et par l'introduction dans l'architecture du système de prototypage d'un module de gestion des accès aux données de la base de données.

Nous pouvons envisager l'analyse de l'impact de cette extension sur les objectifs du système de prototypage, à savoir principalement la fonction d'exécution d'un prototype et la fonction de génération de code final.

4.5.1 Exécution d'un prototype

L'objectif d'exécution et de test d'un prototype du système d'information par exécution de ses spécifications est en partie réalisé par la modification de la structure et de la gestion de la mémoire du système. Le système de gestion des occurrences des types d'objets sous forme de listes chaînées en mémoire centrale a été remplacé par un module de gestion des occurrences sous une base de données relationnelle. Cette gestion permet au système de travailler avec un volume d'information beaucoup plus important, et de montrer à l'utilisateur les performances réelles du système lors de l'accès aux données.

La gestion du système sous une base de données relationnelle va permettre l'introduction d'un mécanisme,la gestion de la notion de transaction, qui rend possible l'exécution du prototype dans un environnement multi-utilisateurs et la prise en charge des problèmes de sécurité et de fiabilité pendant l'exécution. Ce point est discuté dans le chapitre suivant.

4.5.2 Génération de code final

La gestion du système de prototypage sous une base de données relationnelle ne permet pas de réaliser entièrement l'objectif d'évolution du système vers la génération de code final. Mais il constitue un premier pas vers sa réalisation. En effet, le système dispose actuellement d'une mémoire extensible dont les accès sont gérés par un module de gestion. Lors de la génération du code final, les primitives d'accès aux données, définies dans le module de gestion, pourront être traduites de manière automatique en primitives du système final.

4.6 BIBLIOGRAPHIE

- [BODART] BODART F. et PIGNEUR Y. :
  " Conception assistée des applications informatiques:
  Tome 1: Etude d'Opportunité et Analyse Fonctionnelle "
  Masson 1983

- [HAINAUT] HAINAUT J.L. :
  " Conception assistée des applications informatiques:
  Tome 2: Conception de la Base de Données " Masson 1986

CHAPITRE 5 : PRISE EN CHARGE DE LA NOTION DE TRANSACTION

5.1 Objectifs

L'évolution du système de prototypage DSL-PROTO basé sur la génération automatique et l'exécution d'une maquette du système d'information, vers un système de prototypage caractérisé par la production automatique et l'expérimentation d'un prototype du système nécessite de lever les contraintes qui sont imposées par l'exécution d'un modèle réduit, d'une maquette du système.

La restructuration de la mémoire du système en base de données relationnelle et l'introduction d'un module de gestion de la base de données dans l'architecture de DSL-PROTO ont permis de lever la contrainte de gestion d'un volume reduit de données en mémoire centrale. La base de données donne en effet le moyen de stocker et de gérer un volume plus important de données. D'autre part, dans le but de réalisation de l'objectif de reproduction du comportement réel du système, la structure de la base de données permet une organisation et une gestion des données plus proche de la réalité que celle offerte par la mémoire centrale. La réalisation et l'analyse de l'impact de cette restructuration est présentée au chapitre précédent.

L'évolution du système de prototypage s'exécutant dans un environnement mono-utilisateur vers un système de prototypage s'exécutant dans un environnement multi-utilisateur peut se réaliser par l'introduction dans le système d'un mécanisme qui donne la possibilité à plusieurs utilisateurs d'effectuer des traitements sur les mêmes données tout en leur garantissant la bonne exécution de leurs tâches et la cohérence des résultats des traitements effectués sur les données. Ce mécanisme doit donner la possibilité à plusieurs utilisateurs d'expérimenter le prototype du système en travaillant sur les mêmes données enregistrées dans les tables de la base de données dans des conditions identiques à celles dans lesquelles s'exécuterait le prototype dans un environnement mono-utilisateur. Le mécanisme qui rend possible l'exécution du prototype du système d'information dans un environnement multi-utilisateur est

l'introduction et la gestion de la notion de transaction par le système de prototypage. Une transaction est une unité de travail sur la base de données qui fournit le moyen de gérer les conflits d'accès aux données et de garantir la cohérence des données et des résultats des traitements effectués sur ces données par chaque utilisateur.

La troisième contrainte imposée par l'exécution d'un modèle réduit du système d'information est la non-considération des problèmes de sécurité et de fiabilité à l'exécution. Cela signifie que pendant l'exécution de la maquette, le problème des copies de sécurité et de la gestion des points de reprise n'est pas pris en considération. Or il est important de pouvoir garantir à l'utilisateur la cohérence de l'ensemble des données qu'il manipule, de pouvoir assurer que les résultats produits par l'ensemble des traitements effectués sur ces données sont corrects et de maintenir les données dans un état cohérent en fonction des contraintes qui leur sont imposées.

Une dernière contrainte impose de ne pas prendre en compte la recherche de performances du système à l'exécution. Cette contrainte n'est pas une caratéristique typique à l'exécution de la maquette, elle peut également être imposée comme contrainte lors de l'exécution du prototype du système. Mais elle doit évidemment être prise en compte dans le cas où ce sont les performances du système qui font l'objet du prototypage.

La mise en place, les mécanismes à mettre en oeuvre et l'impact de l'introduction de la notion de transaction dans le système de prototypage sont analysées dans les sections suivantes.

5.2 Introduction et définition de la notion de transaction

Deux considérations sont à l'origine de la prise en charge de la notion de transaction. D'une part, l'exécution du prototype requiert le respect des contraintes d'intégrité définies sur les données. Chaque utilisateur de la base de données la reçoit dans un état cohérent et doit la rendre dans un état cohérent. Une base de données est dans un état cohérent lorsque toutes les contraintes d'intégrité définies sur ses données sont satisfaites.
D'autre part, dans la plupart des applications, plusieurs utilisateurs accèdent à la base de données en même temps. Les problèmes peuvent apparaître quand ils désirent effectuer les mêmes types d'opérations sur les mêmes données. Le système de gestion de base de données doit permettre à plusieurs utilisateurs d'accéder à la base de données en évitant les conflits d'accès aux données. Dans le but de satisfaire ces deux contraintes, chaque utilisateur de la base de données doit identifier une unité de travail sur la base de données. Cette unité de travail est appelée transaction.

Une transaction est une opération sur la base de données qui doit s'exécuter comme une entité, c'est-à-dire se terminer entièrement ou ne pas se terminer. Une opération sur la base de données correspond à l'exécution d'une ou plusieurs primitives de manipulation ou de définition de données. Une transaction est caractérisée par la propriété suivante: en début d'exécution elle reçoit la base de données dans un état cohérent et, en fin d'exécution, elle la rend dans un état cohérent. Pour satisfaire cette propriété, une transaction doit, dans certains cas, regrouper plusieurs primitives de manipulation ou de définitions de données.

Considérons l'exemple suivant [GUIDE]:

Un nouveau département est créé avec un nouveau directeur. Une contrainte impose que tout département ait un directeur.
Les tâches suivantes doivent donc être exécutées:

- créer un nouveau département,
- promouvoir un employé au titre de directeur de ce département.

Si une erreur de logiciel ou une défaillance matérielle se produit avant que toutes ces tâches aient été exécutées, la base de données ne sera plus dans un état cohérent. De plus, une exécution partielle entrainerait des erreurs dans les données: un

employé appartiendrait à deux départements et un département n'aurait pas de directeur.
Pour éviter de telles incohérences, toutes ces tâches doivent être exécutées d'un seul bloc, et sont inclues dans une seule transaction.

La portée de la transaction est identifiée par les instructions d'ouverture et de fermeture de transaction. Soit toutes les instructions sont exécutées, soit aucune ne l'est. Dans le cas où la transaction n'a pas été explicitement identifiée par les primitives d'ouverture et de fermeture, le système de gestion de base de données identifie par défaut une transaction à chaque primitive à exécuter.

↳ donc il pourra y avoir incohérence des données

5.3 Introduction de la notion de transaction dans DSL-PROTO

L'introduction de la notion de transaction dans DSL-PROTO est nécessaire pour pouvoir travailler dans un environnement multi-utilisateur en garantissant la cohérence des données.

La notion de transaction est un concept pris en charge par le système de gestion de base de données. Elle est identifiée par les primitives d'ouverture et de fermeture de transaction qui sont mis à disposition de l'utilisateur. Elle est prise en charge de deux manières:

- soit elle est explicitement identifiée,
- soit elle ne l'est pas, et le système l'identifie par défaut à chaque primitive de manipulation ou de définition de données à exécuter.

Ces deux types de prise en charge ne sont pas applicables dans le contexte du système de prototypage.
L'identification explicite nécessite l'intervention de l'utilisateur dans l'identification de la transaction, et sa prise en charge au niveau du langage de spécification ou du langage de prototypage. Or il est important que la notion de transaction reste transparente pour l'utilisateur.
L'identification par défaut n'est pas acceptable, parce qu'elle ne permet pas de garantir à chaque utilisateur la cohérence des données.
Il est donc nécessaire d'introduire la notion de transaction au niveau du système de prototypage DSL-PROTO, et de déterminer un mécanisme qui permet son identification et sa prise en charge de manière automatique.

5.4 Détermination du niveau d'introduction

Une transaction est une unité de travail sur la base de données qui regroupe une ou plusieurs primitives de manipulation ou de définition de données, qui doivent être toutes exécutées, ou aucune ne doit l'être.
Pour que le système DSL-PROTO puisse la prendre en charge de manière automatique, il doit pouvoir l'identifier à des concepts connus et pris en charge par le système. La notion DSL qui correspond le mieux à la notion de transaction est le processus. Celui-ci correspond à l'exécution d'une procédure de traitement de l'information. Les processus définis dans les spécifications du système d'information correspondent aux traitements à effectuer sur les données du système. Le modèle de structuration des traitements, décrit dans [BODART], fournit des critères de décomposition et d'identification de processus. Le modèle propose une décomposition en quatre niveaux, projet, application, phase et fonction.

Le problème engendré par l'introduction de la notion de transaction dans le système de prototypage est de déterminer à quel niveau de décomposition elle doit être introduite, et à quel type de processus elle doit être identifiée.
Deux possibilités peuvent être envisagées. L'introduction de la notion de transaction au niveau phase ou au niveau fonction.

Rappelons la définition d'un processus de niveau phase [BODART]:
Une phase est un traitement possédant une unité spatio-temporelle d'exécution qui implique que celle-ci soit entièrement exécutée dans un centre d'activité homogène dans le temps et dans l'espace, doté de ressources humaines et/ou matérielles et pourvu de règles de comportement nécessaires à son fonctionnement. Cela implique que lors de l'exécution de la phase, il y ait absence de changement spatial dans l'organisation et absence de changement de ressources, que le déroulement de l'exécution puisse avoir lieu sans interruption, c'est-à-dire sans point d'attente et de point de décision, que l'on ait une même périodicité d'exécution, et une disponibilité de toutes les informations au moment du déclenchement.

En tant qu'unité spatio-temporelle d'exécution, une phase a un contexte de cohérence autonome par rapport aux autres phases,

en ce sens qu'elle reçoit en début d'exécution la base de données dans un état cohérent et qu'elle la rend en fin d'exécution dans un état cohérent par rapport aux contraintes d'intégrité définies.
L'introduction de la notion de transaction au niveau phase implique l'identification d'une transaction pour chaque processus de niveau phase à exécuter. Son identification consiste à exécuter une primitive d'ouverture de transaction en précisant le type d'accès qu'elle effectue sur les données et le type d'accès accordé aux autres transactions. Si la transaction bloque la base de données en interdisant son accès aux autres transactions, la base sera bloquée pendant toute la durée de l'exécution du processus de niveau phase. La base de données sera bloquée même lorsque le processus n'effectue aucune manipulation de données. Les implications d'un tel blocage lors de l'exécution en environnement multi-utilisateur sont importantes. Certaines phases risquent de devoir attendre longtemps avant de pouvoir accéder à la base de données.
L'introduction de la notion de transaction au niveau phase n'est donc pas entièrement satisfaisante du point de vue de la gestion des conflits d'accès. Elle doit donc être envisagée au niveau fonction.

Un processus de niveau fonction est associé à un objectif et à un comportement organisationnel considéré comme élémentaire par l'organisation [BODART]. Il résulte de la décomposition d'une phase en sous-problèmes. Une fonction spécifie les règles de traitements relatives à la production de messages et aux actions sur la mémoire du système d'information. L'identification d'une transaction à une fonction permettrait d'affiner les critères d'identification des processus de niveau fonction. Outre les critères exposés dans [BODART], un critère supplémentaire d'identification d'une fonction serait: une fonction constitue un contexte local de cohérence, en ce sens que les contraintes définies sur les données q'elle manipule sont vérifiées en fin d'exécution.

Un transaction ne peut pas être identifiée de manière systématique à un processus de niveau fonction parce que celui-ci ne constitue pas forcément un contexte de cohérence. Il est donc nécessaire d'effectuer un regroupement des fonctions de manière à obtenir un contexte de cohérence. Une transaction peut alors être identifiée à chaque regroupement obtenu, qui forme un contexte de cohérence par rapport aux autres regroupement de fonctions. Les mécanismes à mettre en oeuvre pour l'identification et la prise en charge sont présenté au paragraphe suivant.
L'introduction de la notion de transaction au niveau fonction permet donc la gestion de la cohérence des données du point de vue du respect des containtes d'intégrité définies. Elle permet de satisfaire la propriété suivante: à l'ouverture de la transaction, la base de données est recue dans un état cohérent, et en fin d'exécution, elle est rendue dans un état cohérent.

Nous avons montré qu'au niveau des processus phase, il existe également un contexte de cohérence autonome, dont le système doit tenir compte lors de l'exécution du prototype. Nous proposons donc de considérer un deuxième niveau d'introduction de la notion de transaction. Celle-ci peut être considérée comme une transaction logique qui garantit la cohérence de la dynamique de l'exécution du prototype. Une phase doit entièrement se terminer pour en déclencher une autre. L'enchainement dynamique des phases n'est réalisé que si toutes les contraintes d'intégrité sont vérifiées. A la terminaison de chaque phase, le système contrôle que toutes les contraintes d'intégrité définies sur les données manipulées par la phase sont vérifiées, ce qui devrait être le cas, puisqu'elles sont satisafites au niveau des transactions introduites au niveau fonction.

Les transactions introduites au niveau fonction peuvent être classées comme transactions physiques. Elles correspondent aux transactions prises en charge par le système de gestion de base de données, identifiées par les primitives d'ouverture et de fermeture de transaction. Elles permettent de gérer le plus efficacement possible les conflits d'accès à la base de données, et de garantir la cohérence des données recues par chaque transaction.

Les transactions introduites au niveau phase peuvent être classées comme transactions logiques. Elles définissent un contexte de cohérence autonome garantissant la cohérence de la dyanmique à l'exécution. Exécuter un prototype consiste donc à exécuter une ou plusieurs transactions logiques. Chaque transaction logique correspondant à une ou plusieurs transactions physiques.

Les mécanismes à mettre en oeuvre pour l'identification et la détermination des caractéristiques de ces deux types de transaction sont présentés au paragraphe suivant.

5.5 Mécanismes à mettre en oeuvre

L'introduction de la notion de transaction dans le système de prototypage nécessite lors de l'exécution d'un prototype du système d'information la mise en oeuvre de mécanismes d'identification et de prise en charge de la transaction. Ce n'est en effet que lors de l'exécution du prototype que le système pourra identifier les transactions, logiques et physiques, et déterminer les caractéristiques de la transaction physique. Celle-ci est caractérisée par son type, qui spécifie le type d'accès aux données qu'elle va effectuer, lecture ou mise-à-jour, et par le type de blocage, qui spécifie le type d'accès accordé aux autres transactions sur les mêmes données.

L'introduction de la notion de transaction logique et physique nécessite la mise en oeuvre de respectivement un et deux mécanismes lors de l'exécution du prototype.
En ce qui concerne la transaction physique, les deux mécanismes à mettre en oeuvre sont les suivants: le mécanisme d'identification et le mécanisme de détermination de ses caractéristiques. En ce qui concerne la transaction logique, seul le mécanisme d'identification doit être mis en oeuvre.

5.5.1 Mécanisme d'identification d'une transaction

Le mécanisme consiste à identifier à l'exécution du prototype la portée de la transaction, c'est-à-dire à déterminer le ou les processus qui la forme.
En ce qui concerne la transaction logique, l'identification est immédiate pour chaque processus de niveau phase. Actuellement, le concept de phase est connu du système de prototypage mais il n'est pas pris en charge. Nous pouvons supposer que l'évolution du système se fera dans le sens de cette prise en charge du concept de phase défini en tant qu'attribut du processus.
Une phase est définie par l'ensemble de ses événements déclencheurs et par l'ensemble de ses événements terminaux, qui sont définis dans les spécifications du processus. Lors de la génération de l'environnement statique du prototype, outre les tables de la statique, le système créera la table de définition

des processus de niveau phase, dans laquelle sera enregistré pour chaque phase, l'ensemble de ses événements déclencheurs et l'ensemble de ses événements terminaux. Le système pourra accéder à la table en donnant en entrée un événement déclencheur ou un événement terminal, et la table fournira le nom du processus de niveau phase correspondant.

En ce qui concerne la transaction physique, nous pouvons faire remarquer que le système n'exécute que des processus de niveau fonction. D'autre part, chaque ouverture d'une transaction logique entrainera l'ouverture d'une transaction physique, et la fermeture d'une transaction physique pourra entrainer la fermeture d'une transaction logique. Les deux mécanismes d'identification sont donc fortement liés.
Nous proposons de mettre en oeuvre le mécanisme d'identification suivant:

A l'exécution du prototype, le système vérifie que l'événement déclencheur correspond à un événement déclencheur d'un processus de niveau phase.
Si ce n'est pas le cas, un message d'erreur est envoyé à l'utilisateur.
Dans l'affirmative, un transaction logique est ouverte. Le système détermine le processus de niveau fonction déclenché, détermine les caractéristiques de la transaction à ouvrir et ouvre une transaction physique.
A la fin de l'exécution de chaque processus de niveau fonction, le système détermine si la cohérence des données est réalisée, si les contraintes d'intégrité définies sur les données manipulées sont satisafites. Si ce n'est pas le cas, la transaction physique ne peut pas être fermée et le système passe à l'exécution du processus suivant. Si c'est le cas, la transaction physique peut être fermée. Le système doit alors déterminer si la transaction logique doit elle aussi être fermée. Elle devra l'être si l'événement terminal du processus de niveau fonction qui vient d'être exécuté correspond à un événement terminal du processus de niveau phase qui identifie la transaction logique. Si celle-ci doit être fermée, le système revérifiera que toutes les contraintes d'intégrité définies sur les données manipulées sont satisfaites.

A ce mécanisme d'identification doit être associé le mécanisme de détermination des caractéristiques de la transaction. Celui-ci est présenté au paragraphe suivant.

5.5.2 Mécanisme de détermination des caractéristiques

Une transaction physique est identifiée en appelant le module d'ouverture de transaction auquel le système passe comme paramètres les caractéristiques de la transaction.
Une transaction est caractérisée par son type, c'est-à-dire par le type d'accès, lecture ou mise-à-jour, qu'elle va effectuer sur les données, et par le type d'accès accordé aux autres transactions sur les mêmes données. Pour déterminer ces caractéristiques, le système se base sur la description statique du processus de niveau fonction, qui est enregistrée dans les tables de la statique. Il peut ainsi établir la liste des types d'objets référencés par le processus et la manière dont ils sont référencés. Les types d'objets peuvent être référencés pour une lecture, un ajout, une modification ou une suppression. Les types d'objets qui sont manipulés par le processus sont soit référencés directement dans la section de définition du processus, soit ils sont définis comme sous-ensemble des types d'objets et c'est ce sous-ensemble qui est référencé par le processus.

- Détermination du type de la transaction.

  Si aucun type d'objet n'est référencé pour un ajout, une modification ou une suppression, mais si tous les types d'objets sont référencés pour une lecture, la transaction sera de type LECTURE. Une transaction de type LECTURE est une transaction pour laquelle le système travaille sur une copie de la base de données, pour laquelle les autres utilisateurs peuvent accéder aux mêmes données et pour laquelle les modifications effectuées sur les données par d'autres utilisateurs ne sont pas visibles.

  Si au moins un type d'objet est référencé pour un ajout, une modification ou une suppression, la transaction est de type LECTURE/ECRITURE. Une transaction de type LECTURE/ECRITURE est une transaction qui doit être définie lorsque le processus accède à un type d'objet pour un ajout, une modification ou une suppression dans la base de données. Les types d'objets référencés sont explicitement nommés.

- Détermination du type de blocage.

  Les types d'objets qui sont bloqués pendant l'exécution

de la transaction sont ceux qui sont référencés par le processus. Le type de blocage détermine la permission d'accès accordée aux autres utilisateurs qui désirent travailler sur les mêmes types d'objets. Dans le but de rester le plus général possible et de permettre de determiner facilement le type de blocage, la règle suivante peut être appliquée dans tous les cas:
si la transaction est de type LECTURE, le type de blocage sera LECTURE-PARTAGEE. Ce type de blocage permet à d'autres transactions de type LECTURE d'accéder aux mêmes types d'objets.
Si la transaction est de type LECTURE/ECRITURE, le type de blocage sera ECRITURE-PROTEGEE. Ce type de blocage assure qu'aucune autre transaction ne peut mettre à jour les mêmes types d'objets. Les autres transactions ne peuvent effectuer qu'une lecture sur ce type d'objet.

Nous avons montré qu'une transaction physique doit dans certains cas comprendre plusieurs processus de niveau fonction. Nous tenons à faire remarquer que l'analyse de la description statique du premier processus qui forme la transaction suffit pour déterminer le type de la transaction. En effet, si ce processus ne référence des types d'objets que pour une lecture, il identifiera à lui seul une transaction qui sera de type LECTURE. Si par contre, il référence des types d'objets pour un ajout, une modification ou une suppression, le type de la transaction sera LECTURE/ECRITURE, quelque soit le nombre de processus qui la formera. La transaction sera fermée lorsque toutes les contraintes définies sur les données seront vérifiées.

5.6 Impact de l'introduction de la notion de transaction

L'introduction et la gestion de la notion de transaction dans le système de prototypage DSL-PROTO est un deuxième pas vers la réalisation de l'objectif d'évolution vers un système de prototypage basé sur la génération automatique et l'expérimentation d'un prototype du système d'information. Le premier pas ayant été réalisé par la transformation du système de gestion des donnés en système de gestion de base de données relationnelle.

Le stockage et la gestion d'un volume important de données permet de donner à l'utilisateur une vue plus réaliste des performances du système. La structuration de la mémoire en base de données relationnelle permet d'envisager l'exécution d'un prototype du système dans un environnement multi-utilisateurs. Un premier pas vers la réalisation de cet objectif a été fait en introduisant dans le système la notion de transaction et les mécanismes qui permettent de la gérer.

L'objectif principal de l'introduction de la notion de transaction est la prise en charge des problèmes de cohérence des données à l'exécution du prototype, c'est-à-dire des problèmes de sécurité et de fiabilité des données.
La prise en charge de la notion de transaction garantit d'une part aux autres transactions la cohérence des données qu'elles recoivent et qu'elles vont manipuler, et contribue à garantir l'exactitude des résultats des traitements effectués sur ces données.
D'autre part, elle permet d'envisager l'exécution d'un prototype du système dans un environnement multi-utilisateurs. En effet, un système de gestion de base de données permet à plusieurs utilisateurs d'accéder à la base de données en même temps. La notion de transaction permet de définir pour chaque utilisateur une unité de travail sur la base de données qui permet le contrôle de leurs tâches dans le but d'éviter les conflits d'accès.

L'introduction de la notion de transaction au niveau du système de prototypage est apparue comme une nécessité, étant donné premièrement la restructuration de la mémoire du système en base de données realtionnelle, ce qui a permis d'envisager l'exécution en environnement multi-utilisateur, et deuxièmement l'impossibilité de prendre en charge cette notion au niveau des spécifications du système d'information ou des procédures de prototypage, ni au niveau du système de gestion de base de données relationnelle.

5.7 BIBLIOGRAPHIE

- [BODART] F. Bodart, Y. Pigneur :
  "Conception assistée des applications informatiques
  Tome 1: étude d'opportunité et analyse fonctionnelle."
  Masson 1983.

CHAPITRE 6: CONCLUSIONS

6.1 Evolution des objectifs de l'outil de prototypage

Rappelons les objectifs du système de prototypage DSL-PROTO:

- fournir à l'utilisateur une vue globale du comportement du système d'information pendant son processus de développement,
- permettre une validation expérimentale des spécifications du système d'information dans le but de les vérifier et de les améliorer,
- fournir aux utilisateurs un outil d'apprentissage des règles d'utilisation du système d'information.

Le système de prototypage DSL-PROTO apparait comme un outil mis à disposition des utilisateurs pour expérimenter les spécifications et le comportement du système d'information pendant son processus de développement ainsi que pour l'apprentissage des règles d'utilisation du système.

Dans cette optique, le système de prototypage peut être vu comme un système d'exécution des spécifications basé sur la génération automatique et l'expérimentation d'un prototype du système d'information.

Or les restrictions imposées par l'environnement de travail actuel, à savoir la création et l'exécution d'une maquette, d'un modèle réduit du système d'information, représentent une contrainte importante à la réalisation de cet objectif.
Dans le but de lever cette contrainte, un certain nombre de modifications sont a apporter au système de prototypage.

Un premier pas vers la réalisation de l'objectif peut être franchi en transformant la structure de la mémoire du système.

Elle a été remplacée par une base de données relationnelle dans laquelle sont enregistrées les occurrences des types d'entités et d'associations manipulés par les processus. Un interface sous forme de module de gestion des accès a été introduit dans l'architecture du système pour gérer les accès aux données enregistrées dans la base de données relationnelle.

Pour garantir aux utilisateurs la cohérence des données qu'ils manipulent et pour gérer les problèmes de conflits d'accés, la notion de transaction a été introduite dans le système de prototypage. Une transaction est une unité de travail sur la base de données qui regroupe un ou plusieurs processus de niveau fonction, et qui constitue une entité qui doit être exécutée entièrement ou pas du tout.

La génération automatique du prototype du système d'information consiste à produire automatiquement un maximum d'éléments du programme de prototypage à partir des spécifications du système. Les spécifications comprennent les spécifications des données et des traitements, ainsi que les procédures de prototypage qui réalisent ces traitements. Ces procédures de prototypage sont écrites en langage PROTO et traduites en programmes exécutables. Le code de ces programmes, généré pour l'exécution du prototype est considéré actuellement comme du code-à-jeter, ce qui signifie qu'il n'est pas utilisé dans le système final.
Puisque le code des procédures correspondant aux traitements a été généré pour l'exécution du prototype et qu'il a montré une certaine efficacité , ne pourrait-on pas l'utiliser en tout ou en partie comme composant du système final?
Dans cette optique, le système de prototypage DSL-PROTO devient un système de génération de code final, un générateur automatique d'applications basé sur les spécifications du système d'information.
La gestion du système de prototypage sous une base de données relationnelle est un premier pas vers la réalisation de cet objectif.
Un deuxième pas serait la prise en charge du parallélisme qui peut apparaitre dans les spécifications, et la prise en charge de la réalisation des points de synchronisation.

6.2 Perspectives d'avenir

Le système de prototypage DSL-PROTO apparait comme un bon exemple d'outil de prototypage, mais les restrictions imposées par la création et l'exécution d'un modèle réduit, d'une maquette du système, limite son champ d'action et ses possibilités. Les extensions proposées, à savoir la gestion sous une base de données relationnelle et la prise en charge de la notion de transaction, permettent de faire un pas en avant vers l'obtention d'un vrai système de prototypage basé sur la production et l'expérimentation d'un prototype du système par exécution des spécifications.

Nous avons présenté succintement au chapitre 3 de ce mémoire quelques modifications et extensions possibles du système de prototypage, et ce en réponse à un certain nombre de remarques et de critiques émises.
La prise en charge de ces extensions permettrait de mettre à disposition de l'utilisateur un environnement de travail complet pour l'expérimentation d'un prototype du système d'information pendant son processus de développement.

7 TABLE DES MATIERES

1 CHAPITRE 1 : PRESENTATION DU PROTOTYPAGE 8

1.1 Introduction et définitions 8

1.1.1 Motivations du prototypage 8

1.1.1.1 Les limites des méthodes traditionnelles 9
1.1.1.2 La solution du prototypage 10

1.1.2 Définition 11

1.1.2.1 Le terme 'prototytage' 11
1.1.2.2 L'importance du prototypage 12

1.1.3 Evaluation 14

1.1.3.1 Impacts du prototypage 14
1.1.3.2 Evaluation méthodologique 15

1.2 Aspects et niveaux de prototypage 16

1.2.1 Etapes du prototypage 16

1.2.1.1 Sélection fonctionnelle 16
1.2.1.2 Construction 16
1.2.1.3 Evaluation 17
1.2.1.4 Utilisation future 17

1.2.2 Classes de prototypage 17

1.2.2.1 Prototypage exploratoire 18
1.2.2.2 Prototypage expérimental 18
1.2.2.3 Prototypage évolutif 19

1.3 Techniques et outils de prototypage 21

1.3.1 Techniques de prototypage 21

1.3.1.1 Conception modulaire 21
1.3.1.2 Conception du dialogue 21
1.3.1.3 Simulation 22

1.3.2 Outils de prototypage 22

1.4 BIBLIOGRAPHIE 23

2 CHAPITRE 2 : LE PROTOTYPAGE DANS IDA 25

2.1 Situation actuelle 25

2.1.1 Introduction 25
2.1.2 Spécifications des besoins du système d'information 26
2.1.3 Evaluation de la spécification des besoins 26

2.2 Prototypage des performances 28

2.2.1 Introduction 28
2.2.2 Utilisation de l'outil 28
2.2.3 Test de l'efficience du système 30

2.3 Prototypage des règles de traitement 32

2.3.1 Introduction 32
2.3.2 Buts du système de prototypage 32
2.3.3 Présentation de l'architecture de DSL-PROTO 33

2.3.3.1 Description du rôle de chaque module ... 34

2.3.4 Utilisation du système 37
2.3.5 Présentation du langage de prototypage ... 38

2.3.5.1 Phrases DSL utilisables dans un programme maquette 39
2.3.5.2 Langage d'expression des points de synchronisation 39
2.3.5.3 Langage d'expression d'algorithmes 40

2.3.6 Test de l'effectivité 41

2.4 Critiques de l'outil DSL-PROTO 43

2.4.1 Introduction 43
2.4.2 Critique du domaine d'application du prototypage 43
2.4.3 Critiques des objectifs 45

2.4.3.1 Reproduction du comportement réel 45
2.4.3.2 Validation expérimentale 47
2.4.3.3 Apprentissage des règles d'utilisation . 47

2.4.4 Critique de la production du prototype ... 48
2.4.5 Critique de l'évaluation du prototype 51
2.4.6 Critiques de l'utilisation future du prototype 52
2.4.7 Conclusion 53

2.5 BIBLIOGRAPHIE 54

3 CHAPITRE 3 : EXTENSIONS DE DSL-PROTO 55

3.1 Introduction 55

3.2 Réalisation des objectifs 56

3.2.1 Reproduction du comportement réel 56
3.2.2 Langage de prototypage 57

3.3 Evaluation du prototype 59

3.4 Utilisation future 59

3.5 Conclusion 62

3.6 BIBLIOGRAPHIE 63

4 CHAPITRE 4 : GESTION SOUS UNE BASE DE DONNEES RELATIONNELLE 65

4.1 Objectifs 65

4.1.1 Exécution d'un prototype 65
4.1.2 Génération de code final 66

4.2 Définitions 68

4.2.1 Modification de la structure de la mémoire 68

4.2.1.1 Justification du choix de la structure . 68

4.2.2 Définition du module d'accès 69
4.2.3 Définition du contenu de la base de données 69

4.3 Spécifications du module de gestion 72

4.3.1 Introduction 72
4.3.2 Activation des modules 72

4.3.2.1 Activation du module de création 72
4.3.2.2 Activation du module de mise à jour 73

4.3.3 Spécifications du module de création 73

4.3.3.1 Fonction d'initialisation de la structure 73
4.3.3.2 Fonction de création d'une table 74
4.3.3.3 Fonction de création d'une vue 75
4.3.3.4 Fonction de création d'un index 75
4.3.3.5 Fonction de création d'un champ 76

4.3.3.6 Fonction de modification d'une table ... 77
4.3.3.7 Fonction de modification d'une vue 77

4.3.4 Spécifications du module d'interrogation et de mise à jour 78

4.3.4.1 Fonction d'enregistrement 78
4.3.4.2 Fonction de modification 79
4.3.4.3 Fonction de suppression 80
4.3.4.4 Fonction de lecture 80

4.4 Réalisation de l'interface 82

4.4.1 Introduction 82

4.4.1.1 Caractéristiques du SGBD utilisé 82
4.4.1.2 Caractéristiques de la programmation des modules 83

4.4.2 Réalisation du module de création 83

4.4.2.1 Fonction d'initialisation de la structure 83
4.4.2.2 Fonction de création d'une table 84
4.4.2.3 Fonction de création d'une vue 84
4.4.2.4 Fonction de création d'un index 85
4.4.2.5 Fonction de création de champs 85
4.4.2.6 Fonction de modification d'une table ... 86
4.4.2.7 Fonction de modification d'une vue 86

4.4.3 Réalisation des modules de mise à jour et d'interrogation 86

4.4.3.1 Fonction d'enregistrement d'une occurrence 86
4.4.3.2 Fonction de modification d'une occurrence 87
4.4.3.3 Fonction de suppression d'une occurrence 87
4.4.3.4 Fonction de lecture d'une occurrence ... 88

4.5 Impact de la réalisation du module de gestion 89

4.5.1 Exécution d'un prototype 89
4.5.2 Génération de code final 90

4.6 BIBLIOGRAPHIE 91

5 CHAPITRE 5 : PRISE EN CHARGE DE LA NOTION DE TRANSACTION 92

5.1 Objectifs 92

5.2 Introduction et définition de la notion de transaction 94

5.3 Introduction de la notion de transaction dans DSL-PROTO 96

5.4 Détermination du niveau d'introduction 97

5.5 Mécanismes à mettre en oeuvre 100

5.5.1 Mécanisme d'identification d'une transaction 100

5.5.2 Mécanisme de détermination des caractéristiques 102

5.6 Impact de l'introduction de la notion de transaction 104

5.7 BIBLIOGRAPHIE 105

6 CHAPITRE 6: CONCLUSIONS 106

6.1 Evolution des objectifs de l'outil de prototypage 106

6.2 Perspectives d'avenir 108

7 TABLE DES MATIERES 109