UNIVERSITÉ CHEIKH ANTA DIOP DAKAR

THÈSE

Présentée à

l'École Supérieure Polytechnique

(E.S.P)

en vue de l'obtention

du Diplôme de DOCTEUR-INGÉNIEUR

Spécialité INFORMATIQUE

Par

Samba BALDÉ

SUJET :

CONCEPTION ET RÉALISATION D'OUTILS DE SUPPORT POUR L'INGÉNIERIE ORIENTÉE OBJETS DES BESOINS

Soutenue le 15 octobre 1998

devant le Jury composé de :

Président :	Mary Tew NIANE,	Professeur
Membres	Maurice TCHUENTE,	Professeur
	El Hadj Habib NGOM,	Maître de Conférence
	Alex CORENTHIN,	Maître Assistant
	Mouhamed Tidiane SECK,	Maître Assistant

Année 1998

REMERCIEMENTS

Mes remerciements sincères vont

- à M. Oumar SOCK, Directeur de l'École Supérieure Polytechnique (ESP) pour son soutien, ses encouragements et pour sa disponibilité à toute épreuve, malgré ses hautes fonctions ;

- à M. Mary Tew NIANE, Professeur, Directeur de l'UER de Mathématiques et d'Informatique à l'Université de Saint-Louis pour avoir accepté de corriger un premier jet de ce travail et nous honorer en acceptant de présider ce jury ;

- à Mme Colette ROLLAND, Professeur d'Informatique à l'Université Paris I, Directeur du Centre de Recherche en Informatique (CRI), Responsable de la Formation Doctorale en Ingénierie et Bases de Données, pour avoir accepté l'encadrement scientifique à distance de cette thèse, nous avoir permis d'effectuer quelques séjours au CRI ;

- à M. Maurice TCHUENTE, Professeur à l'Université de Dschang, Cameroun pour sa disponibilité et l'honneur qu'il nous témoigne en acceptant de participer au jury ;

- à M. Habib NGOM, Maître de Conférence à l'UCAD, Directeur des Études à l'ESP pour l'encadrement rigoureux et le souci permanent de réussite qu'il nous a toujours manifesté ;

- à M. Alex CORENTHIN, Maître-Assistant, Chef du Département Informatique à l'ESP pour nous avoir tracé le chemin, mis à notre disposition de bonnes conditions de travail pour la préparation de la thèse, et accepté de participer au jury ;

- à Mouhamed Tidiane SECK, Maître-Assistant à l'ESP pour son soutien et ses conseils précieux et ceci depuis nos premiers pas à l'ESP et avoir accepté de participer au jury ;

- à M. Siméon FONGANG, Professeur à l'ESP, Responsable du Laboratoire de Physique de l'Atmosphère (LPA) dont le soutien et les conseils m'ont été précieux pendant toute la préparation de cette thèse ;

- à M. Daouda BADIANE, Maître-Assistant à l'ESP, Responsable Pédagogique de la section DUT Informatique et de la Formation DTSI pour son soutien moral et son aide précieuse ;

- à M. Sidi FARSI, Maître-Assistant à l'ESP, pour ses conseils et son amitié qu'il n'a jamais cessé de témoigner à mon égard ;

- à M. Samuel OUYA, Enseigant chercheur à l'ESP, Docteur ès Mathématiques, pour son amitié et son soutien sans faille.

Nous tenons à remercier M. DIOP SALL, Professeur à l'ESP pour ses multiples conseils et son soutien constant ; nous saluons son sens aigu de la valeur humaine.

Nous remercions également les collègues Christian CLERCIN et Thierry DE VALOIS pour nos discussions fructueuses lors de nos réunions régulières et pour l'ambiance particulièrement agréable dans laquelle s'est déroulée ce travail.

Nous associons à ces remerciements tous nos collègues du Département Informatique, qui ont eu, à nous épauler d'une façon ou d'une autre dans la réalisation de ce travail.

Nous n'oublions pas le personnel administratif et technique du Département Informatique qui a toujours répondu à nos sollicitations.

A [1]CASE can be defined as follows :

1 The philosophy that encourages the standardized capture of the application development process by documenting the application from conception through coding in a reusable, accessible, automated tool ;

2 Any sofware product that helps to implement the preceding philosophy

by Adrienne Tannenbaum

[1] L'acronyme CASE signifie 'Computer Aided Sofware Engineering'

Table des matières

TABLE DES MATIÈRES

[2] N.B. le mot scénario et l'expression 'cas d'utilisation' seront utilisés de façon synonime dans tout le document.

Introduction

La grande avancée de la programmation (OO) pendant ces quelques dernières années a entraîné l'adoption progressive du paradigme objet dans les méthodologies de système d'information (SI) . L'idée est d'exploiter les caractéristiques utiles de ce paradigme, révélées dans la programmation OO pour un meilleur développement pas seulement des méthodes de conception mais aussi des méthodes d'analyse. C'est la raison fondamentale de l'émergence des méthodes d'analyse OO dans le cadre du cycle de vie du développement des SI [Booch 91] [Coad 90] [Rumbaugh 91] [Shlaer 91] [Meyer 90].
De plus les progrès récents dans le domaine de l'ingénierie logicielle assistée par ordinateur, avec la large disponibilité des outils CASE ont aidé non seulement l'automatisation des méthodes d'analyse à devenir une réalité, mais aussi à la simplification et l'accélération du processus d'automatisation.

Le travail de thèse qui nous a été confié s'inscrit dans cette dynamique : il s'agit en effet de concevoir et réaliser un ensemble (d'outils CASE) de support pour une méthode d'ingénierie des besoins. Nous nous appuyons sur la méthode OO O* [Brunet 1993], mais notre étude est généralisable à d'autres méthodes OO.

De façon générale, l'ingénierie des besoins comprend les activités suivantes :

-une activité d'**élicitation** qui implique que l'analyste collecte de l'information sur les problèmes du client ; ceci peut être réalisé de plusieurs façons : interviews et discussions, observations, étude de la documentation disponible etc.. Récemment, la communauté de l'ingénierie des besoins s'est considérablement focalisée sur le concept de scénario (use case). Notre outil d'élicitation sera basé sur l'investigation des scénarios d'utilisation.

-une activité de **modélisation** dans laquelle l'analyste traite l'information collectée pendant l'activité précédente et met en correspondance les descriptions informelles fournies par le client avec des concepts formels dans un langage des besoins ;

- une activité de **vérification** dans laquelle l'analyste détecte des problèmes (contradiction, incohérences ...) dans le document des besoins ;

-une activité de **validation** dans laquelle l'analyste contrôle l'adéquation de ses descriptions en les reformulant au client d'une façon appropriée ;

Notre contribution couvre ces différentes activités. Nous proposons en effet les outils suivants :
-un éliciteur permettant la capture des besoins de l'utilisateur,
-une interface d'aide à la saisie des concepts de la méthode,
- la gestion de la persistance ,
-un prototypeur qui réalise une application partielle à partir des spécifications.

Nous présentons d'abord un état de l'art sur les méthodes d'analyse OO et les outils CASE de support qui leur sont associées au *Chapitre 1.* Ensuite au *chapitre 2* nous faisons un survol des grandes lignes de la méthode O* avant de présenter les différents outils :

- l'éliciteur au *Chapitre 3*
- la gestion de la persistance au *Chapitre 4*
- l'interface d'aide à la saisie des concepts O* au *Chapitre 5*
- le prototypeur au *Chapitre 6*

Chapitre 1 : État de l'art sur les méthodes d'analyse OO et les outils CASE

Nous commençons par présenter quelques méthodes d'analyse existantes fondées sur des approches antérieures (i.e. l'approche cartésienne, puis l'approche systémique) en section 1.1. Plus spécifiquement, cette section inclut l'introduction des divers concepts utilisés dans les méthodes d'analyse existantes fondées sur l'émergeante approche orientée objet. Un survol des outils CASE est présenté en section 1.2. La section aborde toute la gamme des outils existants: depuis les outils traditionnels rigides jusqu'aux noyaux CASE (CASE shells) récents qui permettent la personnalisation de l'outil en fonction de la méthode de développement. Cette section inclut aussi les environnements CASE pour l'intégration d'outils.

1.1. Survol des méthodes d'analyse

L'émergence des méthodes d'analyse dans les années 1970 a eu pour principal objectif de traiter les problèmes rencontrés dans l'analyse des besoins grâce à des techniques de modélisation conceptuelle. Fondamentalement, les méthodes d'analyse existantes peuvent être divisées en catégories en se basant sur les concepts utilisés dans les différentes approches de modélisation : l'*approche cartésienne*, l'*approche systémique* et l'*approche orientée objet*. La figure 1.1 montre les approches de modélisation à travers trois perspectives : la perspective orientée *données*, la perspective orientée *processus* et la perspective orientée *comportement*.

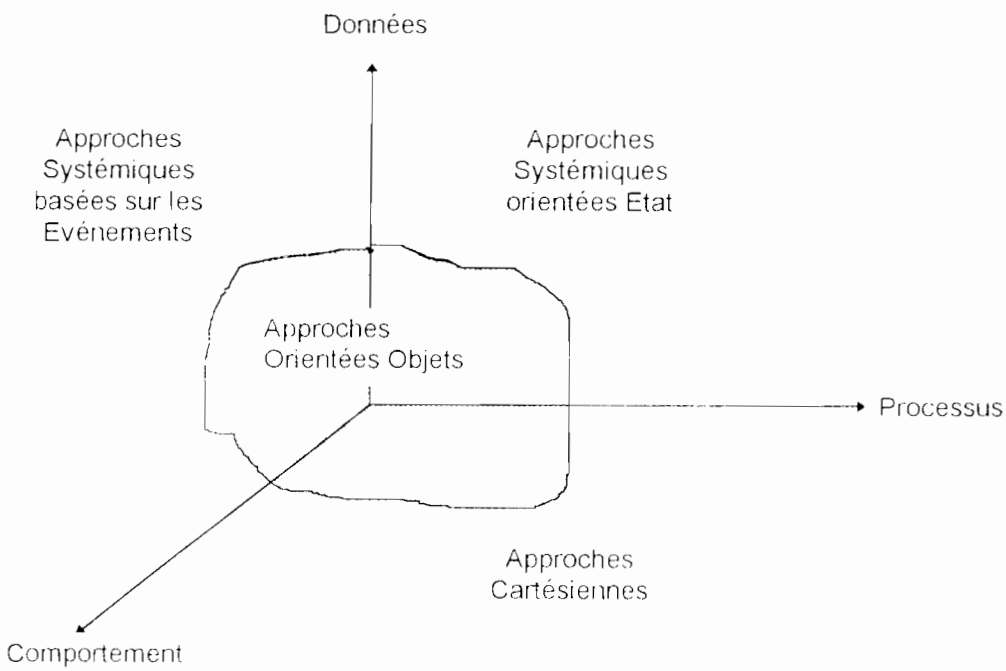

Figure 1.1 Les trois perspectives des approches de modélisation

La perspective orientée données se focalise sur une analyse complète et minutieuse des données et de leurs relations, la perspective orientée processus se focalise sur l'analyse des fonctions/processus des systèmes du monde réel, tandis que la perspective orientée comportement se concentre sur la nature dynamique des données et des événements du monde réel qui ont un impact sur les données enregistrées dans le SI.
Les approches cartésiennes couvrent les perspectives orientées processus et comportement, les approches systémiques basées sur les événements (Event driven Systemic Approaches) couvrent les perspectives orientées données et comportement tandis que les approches systémiques orientées état (State-oriented Systemic Approaches) couvrent les perspectives orientées données et processus. Toutefois, aucune de ces 3 catégories d'approches ne couvrent toutes les 3 perspectives. La zone ombrée sur la figure présente l'émergeante approche orientée objet en tant que grande avancée, en proposant l'intégration des trois perspectives dans de plus en plus de techniques de modélisation conceptuelles courantes.

1.1.1 Méthodes cartésiennes

Les *méthodes cartésiennes* sont issues de l'*intégration du paradigme cartésien et de l'approche de conception fonctionnelle*. La caractéristique dominante dans la procédure d'analyse de ces méthodes est qu'un domaine de problème pour être résolu doit être décomposé hiérarchiquement en fonctions (ou processus). Ces fonctions sont à leurs tours, décomposées par raffinements successifs en de simples sous-fonctions en mettant en oeuvre un principe d'abstraction particulier. La décomposition continue jusqu'au point où les fonctions obtenues sont gérables intellectuellement. Comme point de départ de la décomposition fonctionnelle le système est vu à un haut niveau en termes de ce QU'il est censé réaliser (le QUOI), et ensuite à un niveau de conception détaillée, COMMENT il réalisera ces objectifs orientés processus.

Ces méthodes s'appuient fortement sur les *flots de données*. L'outil le plus connu supportant ces méthodes est le *Diagramme de Flots de Données (DFD)*. Dans un DFD, un nœud peut être une *fonction* (ou un *processus*), un *répertoire de données* ou une *entité externe*, et le lien entre deux nœuds est un flot de données d'un nœud vers un autre. Le lien entre deux sous-fonctions de la même fonction représente un flot d'information, et quelquefois, un flot de contrôle. La figure1.2 modélise des parties d'un domaine bancaire en utilisant un DFD.

Deux exemples bien connus de méthodes d'analyse cartésiennes sont SA (*Structured Analysis*) [Ross 77, De Marco 78] et SADT (*Structured Analysis and Design Technique*) [Ross 77]. L'intégration subséquente de quelques méthodes d'analyse cartésiennes et des environnements de conception OO était due au fait que ces deux méthodes cartésiennes, prises entre autres, ont été largement utilisées à travers le monde entier. La principale motivation était de fournir une méthode de développement OO qui soit

familière à une majorité d'analystes. Cependant, la transformation entre une spécification de type DFD et une spécification OO est considérée comme indirecte et plutôt difficile à réaliser.

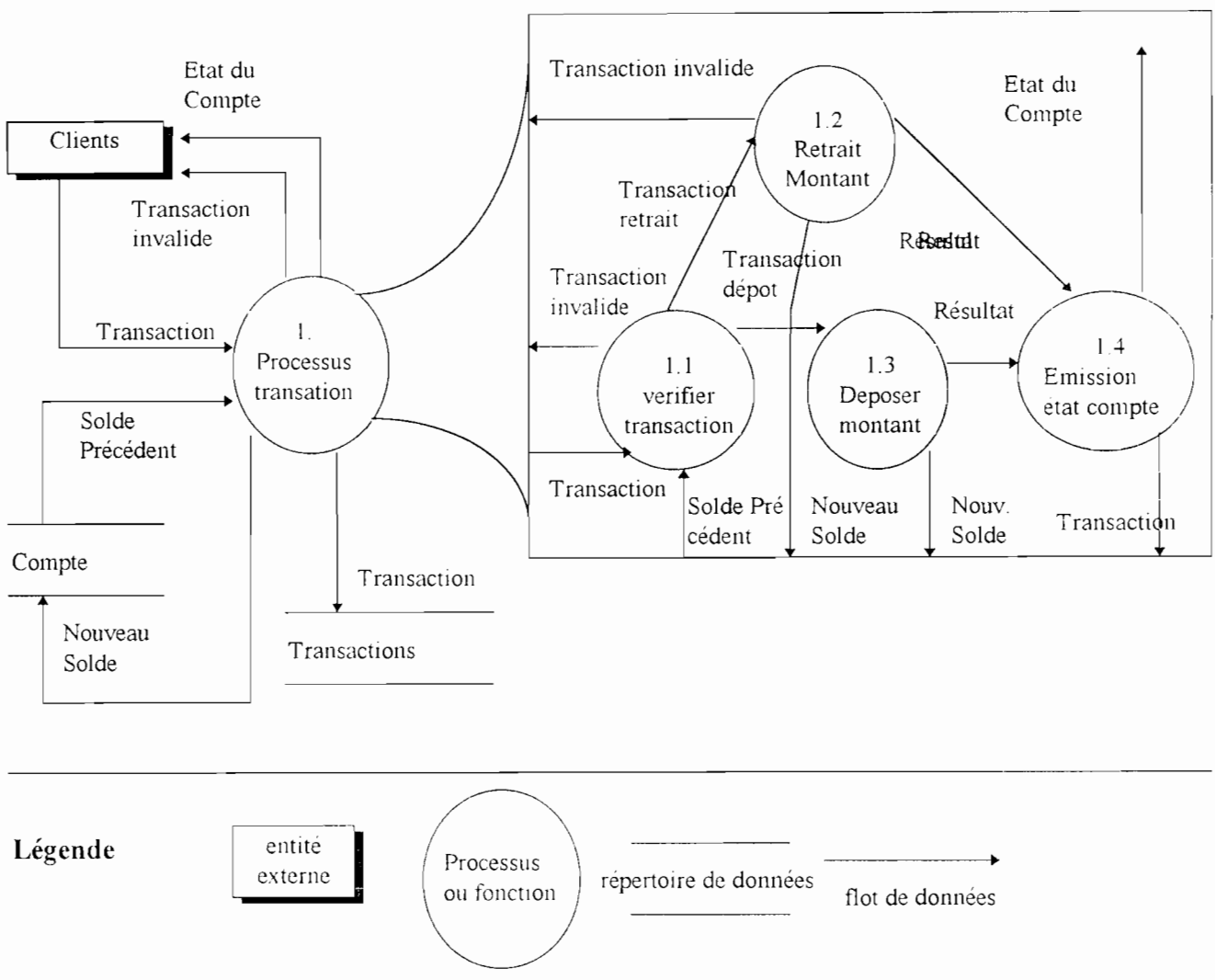

Figure 1.2 Un DFD modélisant des parties d'un domaine bancaire

Pour alléger ce problème. des travaux existants fournissent des heuristiques de transformations partielles, mais souvent, ils nécessitent l'intervention de l'analyste pour quelques prises de décision.

Plusieurs auteurs proposent des variations des DFDs. Bailin [Bailin 89] propose de spécifier des EDFD (*Entity Data Flow Diagrams*), dans lesquels les noeuds du graphe sont des entités ou des fonctions. Une entité est soit active, elle est alors détaillée par un EDFD, soit passive, elle correspond alors à un répertoire de données. Une fonction ne peut être détaillée qu'en sous-fonctions et doit appartenir au contexte d'une entité, en agissant sur elle ou en en étant exécutée par elle. Pour [Henderson90], ces diagrammes peuvent être utilisés en conception OO pour représenter les *liens d'utilisation* entre les objets.

Schiel [Schiel 89] propose les diagrammes de flots d'informations (IFD ou *Information Flow Diagram*), dans lesquels le terme information fait référence au contenu des messages échangés par les objets. La particularité des IFDs est que les données sont représentées par les objets et ne circulent pas dans des flots au sens des DFDs.

Le principal inconvénient des méthodes cartésiennes est l'inadéquation de leurs caractéristiques de modélisation de données pour la représentation de données dans une perspective globale. D'autres désavantages sont liés au manque de travail théorique fournissant une fondation solide aux concepts et techniques de la décomposition descendante et le manque d'aides méthodologiques pour le développement de SIs. De plus, la difficulté liée à l'utilisation des DFDs réside dans l'évaluation de la cohérence, de la complétude et de la qualité des solutions.

1.1.2. Les méthodes systémiques

Les *méthodes systémiques* se focalisent totalement sur la modélisation de données. Elles sont basées sur *l'intégration du paradigme systémique et de l'approche conceptuelle*. Contrairement aux méthodes cartésiennes, l'approche systémique est issue de la *théorie des systèmes* [Lemoigne 77]. Les méthodes d'analyses basées sur cette approche mettentt l'accent sur la représentation de phénomènes pertinents de l'univers d'application à travers un modèle conceptuel. Au lieu d'analyser le monde réel en termes de tâches ou fonctions comme dans l'approche cartésienne, l'approche systémique appréhende la réalité comme un ensemble d'entités ou d'objets lesquels ont des relations et des interactions entre eux et qui évoluent au cours du temps. L'approche systémique a pour but d'amener à comprendre et à interpréter une réalité complexe à travers la modélisation de ses phénomènes pertinents.

Les modèles conceptuels définis dans les méthodes d'analyse systémiques sont fortement influencés par les travaux dans le domaine de l'intelligence artificielle sur les réseaux sémantiques. Ces modèles étaient à l'origine uniquement des modèles de données ; entre autres, les deux plus connus sont le *modèle Entité/Relation* (modèle E/R) [Chen 76] et le modèle NIAM [Verheiyen 82]. Le modèle E/R utilise les 3 concepts : *type d'entité, type de relation* et *type d'attribut* tandis que NIAM propose les deux concepts : l'*entité* et la *relation binaire*. Ces modèles sont classifiés comme des modèles sémantiques parce qu'ils expriment la représentation sémantique de la connaissance.

Dans les modèles E/R, un type d'entité représente un ensemble d'entités de même nature, c'est à dire possédant les mêmes types d'attributs tandis qu'un type de relation (ou type d'association) représente un ensemble de relations de même nature, c'est à dire possédant les mêmes types d'attributs, et définies entre les mêmes types d'entités.

Un type de relation peut être caractérisée par une cardinalité minimum et une cardinalité maximum. La cardinalité minimum (resp. cardinalité maximum) représente le nombre minimum (resp. le nombre maximum) de relations d'un type de relation, dans lesquelles chaque occurrence d'un type d'entité peut être impliquée.

La figure 2.3 décrit le formalisme graphique le plus utilisé. Elle visualise le modèle E/R pour un système élémentaire de commande de produits dans une entreprise.

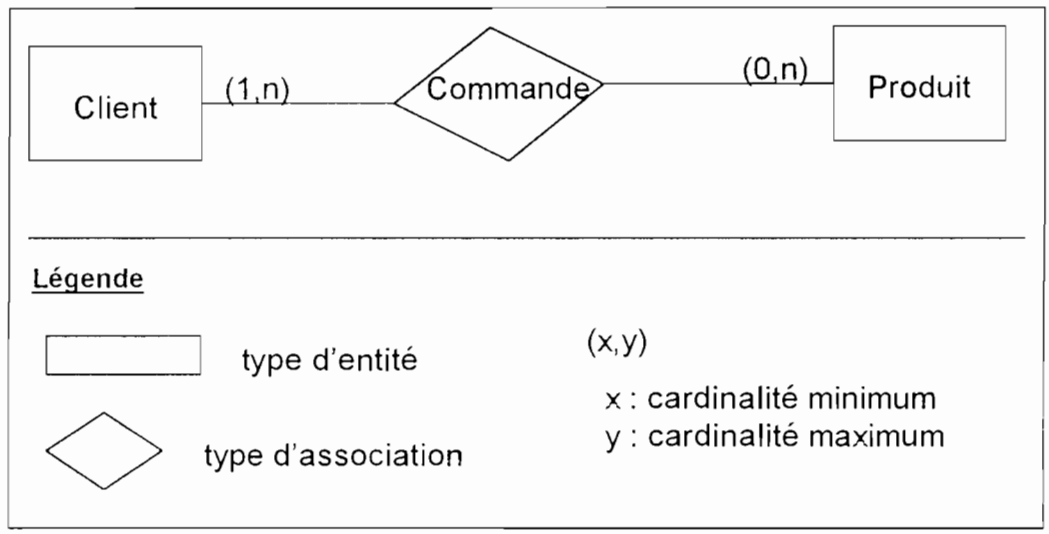

Figure 1.3 : Le schéma E/R la commande d'items de produits

Le trait reliant un type d'entité à un type de relation est appelée *rôle* des occurrences du type d'entité dans les occurrences du type de relation. A chaque rôle est associé une contrainte référentielle, spécifiant que les entités reliées à chaque relation doivent exister. Bien que le modèle E/R est d'utilisation simple, il lui manque la représentation de liens comportementaux des phénomènes du monde réel.

Le modèle NIAM adopte la même perception de la réalité, à l'exception qu'il restreint les associations à des relations binaires entre entités. Dans le modèle de Smith [Smith 77], les types d'entités et les types d'associations sont des objets qui peuvent être structurés par l'utilisation de l'une des trois formes d'abstraction : *classification*, *agrégation* et *généralisation*, résultant ainsi à une hiérarchie d'objets. La classification permet aux objets de même nature d'être groupés en classes, l'agrégation permet de d'interpréter une association entre des objets (appelés *composants*) comme un objet

de plus haut niveau (appelé *agrégat*), tandis que la généralisation permet à un ensemble d'objets (appelés *objets spécialisés*) d'être vus comme un objet complexe unique (appelé *objet générique*).

Due à la limitation mentionnée ci-dessus des méthodes systémiques d'être orientées vers la représentation de données, plusieurs travaux ont visé à intégrer la représentation des aspects dynamiques des phénomènes du monde réel. Cela conduisit à l'émergence de quelques méthodes d'analyse capables d'intégrer les aspects structuraux et comportementaux du SI futur dans le même schéma conceptuel. Quelques-unes des ces premières méthodes furent REMORA [Rolland 82] et ACM/PCM [Brodie 82]. Ces méthodes possèdent une façon commune de modéliser les objets du monde réel : les objets subissent des changements d'états au cours du temps, et pendant leur durée de vie, des changements d'états successifs ont lieu. Les objets possèdent des propriétés structurelles et comportementales. La *structure* d'un objet décrit ses *relations internes et externes* avec d'autres objets. Le *comportement* exprime *quand* et *comment* les objets subissent des changements d'états.

Dans la méthode REMORA, les changements d'états des objets résultent de l'exécution d'opérations. L'exécution d'opérations est le résultat de l'occurrence d'un événement. Un événement se produit instantanément à un moment donné lorsqu'un changement d'état particulier vient juste d'être constaté sur un objet, et a comme conséquence le déclenchement de certaines opérations. Par conséquent, les méthodes orientées comportement mettent l'accent sur l'évolution cohérente des états des objets, la façon dont les objets évoluent au cours du temps et quand les changements d'états sur ces objets ont lieu. Le temps est partie intégrante des modèles conceptuels orientés comportement. Dans la méthode CIAM [Gustafsson 82], les attributs d'un type d'entité dépendant du temps sont définis comme des fonctions du temps. La modélisation du temps fait partie de la définition de la méthode, dans des méthodes telles que ERAE [Dubois 86], TEMPORA [Tempora 89] et OBLOG [Sernadas 89].

1.1.3 Méthodes orientées objet

L'émergence des méthodes orientées objet est principalement due au besoin de l'intégration des trois perspectives de la figure 1.1 dans le schéma conceptuel du SI futur de telle façon que tous les aspects du système du monde réel puissent être entièrement capturés. Ces méthodes sont en partie basées sur les approches évoquées dans les sections précédentes. Les méthodes d'analyse OO existantes utilisent largement les concepts des modèles sémantiques. De plus, des notions telles que *client-serveur* et *lien d'utilisation* mis en oeuvre dans la conception OO ont aussi été adaptés à quelques méthodes d'analyse OO. Nous décrirons brièvement dans la suite quelques concepts importants utilisés par la plupart des modèles conceptuels OO existants, en faisant des références et comparaisons avec les modèles conceptuels obtenus en suivant l'approche cartésienne ou systémique.

. L'objet et la classe

Toutes les méthodes d'analyse OO possèdent une mécanisme d'abstraction permettant de regrouper les objets en classes décrivant certaines de leurs caractéristiques. Dans certaines méthodes, les concepts d'*entité* et de *type d'entité* des modèles E/R traditionnels sont utilisés au lieu des concepts d'objet et de classe. Une des spécificités de ces méthodes est que toutes les spécifications dynamiques pour représenter les interactions des objets sont encapsulées dans les classes. Le concept de *classe* représente le phénomène caractérisé par une collection d'attributs et de méthodes. La principale différence entre le concept d'objet et le concept d'entité est que le concept d'objet possède une sémantique plus riche, il inclut les caractéristiques statiques et dynamiques du phénomène. Le concept d'entité fournit uniquement une vue statique d'un phénomène.

. L'attribut

Dans les langages de programmation OO, la notion *attribut* prend sa valeur dans un type de donnée ou une classe, tandis que la même notion dans certains modèles sémantiques est utilisée pour associer un type de donnée à une classe. Les attributs d'une classe sont définis de façon unique. Les attributs qui prennent leur valeur dans une classe correspondent aux liens d'association ou liens agrégation (définis ci-dessous). Les types de données ou domaines peuvent être des *types prédéfinis* (i.e. **entier**, **réel**, **booléen**, **date**, **chaîne**, etc.), des *types énumérés définis par l'analyste*, ainsi que des *types prédéfinis auxquels un on associe ensemble de règles ou des bornes d'intervalle*. Des attributs complexes sont définis dans la plupart des modèles existants [Spaccapietra 89, Cauvet 89].

. Liens statiques : lien d'association, lien agrégation, lien d'utilisation et lien de généralisation

Les liens statiques permettent de caractériser des relations ayant une certaine durée dans le temps, entre des objets ou entités. Quatre principaux types de liens sont utilisés dans les méthodes d'analyse OO.

. Lien d'Association

Le lien d'*association* s'inspire du concept de relation du modèle E/R. Il est lié à la forme d'abstraction dans laquelle une relation entre des objets membres, est considérée comme un ensemble d'objets de plus haut niveau. Il est utilisé dans les modèles conceptuels tels que OMT [Rumbaugh 91], OOSA [Shlaer 91], PTECH [Fowler 91] et OOA [Coad 90]. Bien que le modèle E/R soit simple et largement utilisé, il présente plusieurs inconvénients pour un passage vers des spécifications OO. Un de ces inconvénients est la confusion dans la transformation des types de relations. En l'occurrence, certains types de relations sur lesquels des caractéristiques dynamiques peuvent être définies se comportent comme des classes, tandis que d'autres types de

relations se comportent comme des attributs de classes. La raison en est qu'il est impossible de modéliser explicitement les aspects comportementaux de phénomènes du monde réel dans le modèle E/R.

. Lien d'agrégation

La littérature sur le lien d'*agrégation* qui apparait dans les modèles sémantiques date de [Smith 77]. Il est lié à la forme d'abstraction dans laquelle une collection d'objets sont vus comme un unique objet de niveau plus élevé. Le lien d'agrégation définit une relation d'appartenance entre un *objet agrégat* et un ou plusieurs *objets composants*. C'est un cas particulier du lien d'association. La sémantique particulière entre deux objets x et y peut être décrite de deux façons : $x <$ *possède un*$> y$, ou réciproquement, $y <$ *est une partie de* $> x$ où x est un objet agrégatif et y est l'objet composant. Par exemple, une voiture est composée d'une carrosserie, d'un moteur et plusieurs roues. En dépit du consensus sur la sémantique du lien d'agrégation, ses propriétés varient de façon importante suivant les approches.

Dans [Srnith 77], l'agrégation est avant tout une abstraction par laquelle une association entre objets est interprétée comme un objet de plus haut niveau. Cette vision peu restrictive permet par exemple de former un objet agrégat «réservation» à partir de la relation existant entre une personne, un hôtel et une date. On trouve la même approche dans le modèle ACM/PCM [Brodier 82] où, dans un exemple de gestion d'une conférence, on modélise un auteur comme agrégat de personne et de papier. Le modèle OMT [Rumbaugh 91] préconise que deux objets ayant une vie indépendante ne doivent être reliés par un lien d'agrégation, mais par un lien d'association. Par exemple, une société est agrégat de ses départements, mais pas de ses employés. Un de ses critères d'identification d'un lien d'agrégation concerne les aspects dynamiques : des opérations de l'objet agrégat peuvent s'appliquer automatiquement à ses objets composants. Un autre critère utilisé est la dissymétrie qui doit exister entre les deux objets, l'objet composant étant subordonné à l'objet agrégat. Le lien d'agrégation est aussi employé dans le modèle OOA [Coad 90], le modèle MADIS [Essing 91] et dans [Booch 91, Henderson 91].

. Lien d'utilisation

Le *lien d'utilisation* a été mis en oeuvre dans les méthodes de conception OO et a été introduit par [Booch 82] avec l'objectif d'aider dans la conception logicielle ADA. La relation d'utilisation s'inspire fortement des principes du génie logiciel. Il devient de plus en plus populaire dans les méthodes d'analyse OO. Le lien d'utilisation [Booch 91] établit un chemin permettant l'envoi de messages d'un objet vers un autre. Sa sémantique est qu'un objet client «utilise» les services d'un objet serveur, ou fournisseur. Ainsi, il est aussi nommé *lien client-serveur* comme dans le modèle OOA [Coad 90]. En conséquence, du fait son support pour l'envoi de message, le lien d'utilisation possède une forte connotation dynamique, bien qu'il soit considéré comme un lien statique. Toutefois, le lien d'utilisation est plus orienté vers l'étape de conception

technique [Teisseire 91]. Une opinion couramment admise [Henderson 91] est de considérer que les liens d'association et d'agrégation utilisés dans l'étape d'analyse se transforment en liens d'utilisation dans l'étape de conception.

. Lien de Généralisation

La forme d'abstraction appelée *généralisation* capture les similarités entre objets, et en même temps ignore les différences entre eux. Le besoin de différencier le concept d'héritage utilisé dans les langages de programmation OO du concept (*généralisation / spécialisation*) utilisé dans les modèles conceptuels OO a été noté par plusieurs auteurs. Par exemple, dans [Castellani 91], *l'héritage par spécialisation* qui est utilisé pour représenter un lien sémantique réel, est différencié de *l'héritage par factorisation* mis en oeuvre pour la réutilisation des caractéristiques des objets. Dans le premier cas, chaque instance d'une classe spécialisée appartient aussi à la classe généralisée, tandis que dans le dernier cas, une instance d'une sous-classe n'appartient pas à la superclasse. Il est possible que la superclasse n'ait pas d'instances, auquel cas, la superclasse sert à définir uniquement des déclarations de méthodes et d'attributs utilisés par les sous-classes terminales dans la hiérarchie d'héritage.

Dans ce contexte, un *mécanisme de liaison tardive* [Wegner 89] est adopté par certains langages de programmation OO pour l'accès aux attributs définis dans la classe généralisée, par un objet d'une classe spécialisée. D'autres concepts voisins sont : le *groupe de spécialisation* (*cluster*) [Smith 77] qui représente un ensemble de classes spécialisées dont les instances sont disjointes ; la *surcharge* (*overriding*) qui dénote le fait qu'un objet spécialisé, au lieu d'exécuter la méthode définie dans la classe généralisée, exécute plutôt la méthode définie dans la classe spécialisée qui possède le même nom que la méthode de la classe généralisée; alternativement, le *mécanisme d'augmentation* (*augmentation mechanism*) [Loomis 87], par lequel la méthode de la classe généralisée est exécutée en plus de la méthode de la classe spécialisée.

La figure suivante présente un synoptique de l'utilisation des différents types de liens statiques dans quelques méthodes d'analyse orientées objet représentatives.

sémantique de l'événement dans laquelle l'événément est considéré au travers de l'évolution des objets. Chaque objet possède un cycle de vie qui consiste en l'ensemble des événements qui l'ont affecté au cours de sa vie. Le concept d'événement ainsi défini est qualifié de déclaratif ou proscriptif. Cette approche est plus proche de la perception que l'on peut avoir de la dynamique du monde réel. Des synchronisations sont aussi proposées par différents mécanismes de partages des événements comme dans le modèles OBLOG et MODWAY.

. Contraintes statiques et dynamiques

Un modèle conceptuel doit être suffisamment riche pour permettre de définir des concepts qui gèrent les contraintes sur toutes les *perspectives possibles*. Dans la littérature, on peut noter que les propriétés stables et les aspects instantanés des SIs modélisés par des concepts de modèles conceptuels doivent être contraints de telle sorte que la cohérence et la fiabilité des SIs puissent être assurées. Cela oblige l'état d'un SI à satisfaire aux contraintes décrites dans un modèle conceptuel donné. Quelques chercheurs considèrent les contraintes comme des règles qui contrôlent le contenu de la base d'information. Toutefois, il n'y a pas encore de standard accepté pour modéliser les contraintes, spécialement les contraintes dynamiques. Par conséquent, ce domaine reste encore un thème continu de recherche.

Contraintes statiques

Les contraintes statiques définies dans une classe sont des contraintes qui doivent être vérifiées sur les objets de la classe à chaque instant, avec comme conséquence, la limitation des états possibles des objets. Par exemple, la contrainte statique du lien d'agrégation définie implicitement entre une voiture et un moteur, stipulant qu'un moteur est toujours associé à une voiture. D'autres contraintes ne sont pas impliquées par les liens statiques, en particulier les contraintes sur les valeurs des attributs (e.g. l'attribut âge de la classe EMPLOYE doit avoir une valeur supérieure à 17), ou les contraintes sur les cardinalités des liens statiques (e.g. le nombre de roues d'une voiture ne peut plus grand que 5). Quelques modèles célèbres (OBLOG [Costa 89], TROLL [Jungclaus 91]) permettent la spécification de telles contraintes.

. Les contraintes dynamiques

Les contraintes dynamiques sont des contraintes qui restreignent l'évolution des objets au cours du temps. Leur description fait intervenir des aspects temporels. Trois approches autorisent la spécification de contraintes dynamiques : l'*approche par graphes de transitions d'état*, l'*approche par la logique modale* et l'*approche par la logique temporelle*. Seule la première est largement répandue.

Un *graphe de transition d'état* peut être défini sur une classe. Il décrit le cycle de vie possible des objets, c'est-à-dire l'ordonnancement des événements qui peuvent les affecter au cours de leur vie. Les nœuds du graphe représentent les états dans

sémantique de l'événement dans laquelle l'événément est considéré au travers de l'évolution des objets. Chaque objet possède un cycle de vie qui consiste en l'ensemble des événements qui l'ont affecté au cours de sa vie. Le concept d'événement ainsi défini est qualifié de déclaratif ou proscriptif. Cette approche est plus proche de la perception que l'on peut avoir de la dynamique du monde réel. Des synchronisations sont aussi proposées par différents mécanismes de partages des événements comme dans le modèles OBLOG et MODWAY.

. Contraintes statiques et dynamiques

Un modèle conceptuel doit être suffisamment riche pour permettre de définir des concepts qui gèrent les contraintes sur toutes les *perspectives possibles*. Dans la littérature, on peut noter que les propriétés stables et les aspects instantanés des SIs modélisés par des concepts de modèles conceptuels doivent être contraints de telle sorte que la cohérence et la fiabilité des SIs puissent être assurées. Cela oblige l'état d'un SI à satisfaire aux contraintes décrites dans un modèle conceptuel donné. Quelques chercheurs considèrent les contraintes comme des règles qui contrôlent le contenu de la base d'information. Toutefois, il n'y a pas encore de standard accepté pour modéliser les contraintes, spécialement les contraintes dynamiques. Par conséquent, ce domaine reste encore un thème continu de recherche.

Contraintes statiques

Les contraintes statiques définies dans une classe sont des contraintes qui doivent être vérifiées sur les objets de la classe à chaque instant, avec comme conséquence, la limitation des états possibles des objets. Par exemple, la contrainte statique du lien d'agrégation définie implicitement entre une voiture et un moteur, stipulant qu'un moteur est toujours associé à une voiture. D'autres contraintes ne sont pas impliquées par les liens statiques, en particulier les contraintes sur les valeurs des attributs (e.g. l'attribut âge de la classe EMPLOYE doit avoir une valeur supérieure à 17), ou les contraintes sur les cardinalités des liens statiques (e.g. le nombre de roues d'une voiture ne peut plus grand que 5). Quelques modèles célèbres (OBLOG [Costa 89], TROLL [Jungclaus 91]) permettent la spécification de telles contraintes.

. Les contraintes dynamiques

Les contraintes dynamiques sont des contraintes qui restreignent l'évolution des objets au cours du temps. Leur description fait intervenir des aspects temporels. Trois approches autorisent la spécification de contraintes dynamiques : l'*approche par graphes de transitions d'état*, l'*approche par la logique modale* et l'*approche par la logique temporelle*. Seule la première est largement répandue.

Un *graphe de transition d'état* peut être défini sur une classe. Il décrit le cycle de vie possible des objets, c'est-à-dire l'ordonnancement des événements qui peuvent les affecter au cours de leur vie. Les nœuds du graphe représentent les états dans

lesquels peuvent être chacun des objets. Les arcs correspondent aux transitions d'un état à un autre que subit l'objet quand il est affecté par un événement. Le modèle de la méthode OMT [Rumbaugh 91] utilise la notion d'*opération* pour décrire comment l'objet réagit en réponse aux événements tandis que le modèle de la méthode OOSA [Shlaer 91] utilise le modèle de MOORE, par lequel des actions ne sont plus associées aux transistors mais aux états.

En *logique temporelle*, deux approches sont à distinguer : l'approche classique et l'approche modale. Dans l'approche classique un intervalle de temps est assigné à chaque assertion. Dans l'approche modale, quelques opérateurs temporels sont utilisés pour déterminer l'instant auquel une assertion peut être évaluée. Une combinaison des deux approches est utilisée dans quelques modèles. Le modèle OBLOG et le modèle TROLL [Jungclauss 91] permettent l'établissement de formules de la logique temporelle du premier ordre en définissant des opérateurs temporels tels que *sometimef()* et *alwaysf()* dont la sémantique est : "à un moment dans le futur" et "toujours dans le futur".

Le modèle OBLOG [Sernadas 89] utilise une technique qui consiste à décrire des dépendances entre événements en utilisant une approche basée sur la logique modale. Cette logique permet de représenter d'une part *ce qu'il est possible (ou impossible) de faire*) et d'autre part *ce qu'il est nécessaire (ou non) de faire*. Chaque objet peut ainsi être caractérisé par des règles de sécurité (*safety*) et par des règles de cycle de vie (*liveness*). Une règle de sécurité se présente sous la forme: *{condition} événement*. Sa sémantique est que l'événement ne peut se produire que si la condition est vraie. La condition peut porter sur les valeurs des attributs. Par exemple un objet "communication" d'un système d'information gérant une conférence possède un statut pouvant prendre les valeurs *soumis*, *accepté* et *rejeté*. Les règles de sécurité : *{status=soumis} acceptation* et *{status = soumis} refus* spécifient que les événements d'*acceptation* et de *refus* ne peuvent affecter le cycle de vie d'un papier que si celui-ci a pour statut *soumis*. Une règle de cycle de vie permet de spécifier qu'un ou plusieurs événements devront nécessairement affecter le cycle de vie de l'objet. Sa syntaxe est celle d'une formule booléenne dont les prédicats sont des événements. Par exemple, la règle de cycle de vie (*acceptation* $\vee$ *refus*) de l'objet "papier" spécifie que l'un ou l'autre de ces événements doit survenir à un moment donné dans la vie du papier.

1.1.4 Guides méthodologiques

Cette section est consacrée aux guides méthodologiques supportant la démarche d'analyse et de spécification d'un schéma conceptuel dans une perspective OO. Il existe de nombreux modèles décrivant le processus d'analyse et de conception d'un SI. Ces modèles sont des représentations abstraites de l'activité de transformation des phénomènes réels et des besoins informels en spécifications conceptuelles, puis internes, du SI. Ce sont des moyens de planification et de contrôle du processus. Il existe des *modèles orientés-activités*, des *modèles orientés-produits* et *des modèles orientés-décisions*, chacun mettant en avant un aspect particulier du processus. Les

modèles orientés-activités représentent le processus de développement comme une succession d'activités de transformation.
Actuellement, les méthodes d'analyse OO ne sont associées qu'à des modèles orientés-activités. C'est pourquoi nous ne présentons pas les autres types de modèle (pour une étude exhaustive voir par exemple [Grosz 91]).

1.1.4.1 - Les modèles "faire et réparer" et "cascade"

Le modèle "faire et réparer" (code and fix) est le plus élémentaire. Il consiste à représenter le processus de développement comme une suite d'itérations d'activités de programmation et de correction d'erreurs. Les problèmes soulevés par cette approche ont engendré l'apparition d'autres modèles de processus orientés-activités, spécialement ceux qui sont constitués d'étapes bien définies.
Le modèle en cascade (Waterfall) recommande (Figure 1.4) que chaque étape soit terminée avant de commencer la suivante. Les inconvénients majeurs de ce modèle sont le manque de flexibilité du processus associé et le fait que les résultats concrets apparaissent très tard par rapport à l'énoncé initial des besoins.

1.1.4.2 Le modèle en spirale

Afin de résoudre les problèmes du modèle en cascade [Boehm 87] propose le modèle en spirale, qui rend explicite l'évaluation des alternatives de conception par la réalisation de prototypes au cours des différentes phases de conception. Ce modèle présente le processus comme la succession des phases suivantes : choix des objectifs, évaluation des alternatives, développement et vérification, planification.
Pour améliorer le modèle en spirale, Iivari [Iivari 90] a introduit le modèle hiérarchique en spirale, qui est une extension du modèle en spirale. Ce modèle supporte trois spirales évoluant indépendamment les unes des autres, chacune des spirales

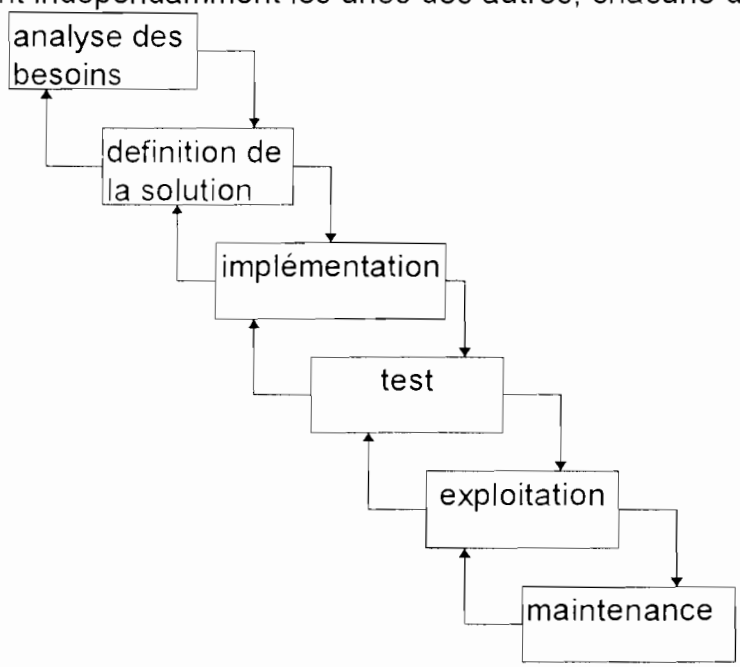

Figure 1.4 : Le modèle en cascade

correspond à un niveau d'abstraction sur le produit : niveau organisationnel, niveau conceptuel et niveau logique.

1.1.4.3 - Le modèle en fontaine

Le modèle en fontaine [Henderson 90] est un autre exemple de modèle orienté-activités. Ce modèle décrit le processus de développement OO par un ensemble de sept activités (Figure 1.5) représentées de bas en haut. Bien qu'il soit peu formel, ce modèle présente l'intérêt de mettre l'accent sur la représentation des recouvrements entre les étapes, ainsi que sur l'itération qui peut amener le développeur à redescendre dans les étapes précédentes, suivant l'analogie du principe d'une fontaine à vases imbriqués. L'idée est que la complétude d'une activité (vase plein) peut amener à passer à l'étape suivante, mais aussi à redescendre vers une activité précédente non terminée (vase non plein) à partir de laquelle le processus reprend.

Le module peut être une classe individuelle ou un groupe de classes. Les classes ou groupes de classes ne sont acceptés dans la librairie qu'après un passage réussi dans toutes les sept étapes. L'étape sept correspond à l'activité d'itération.

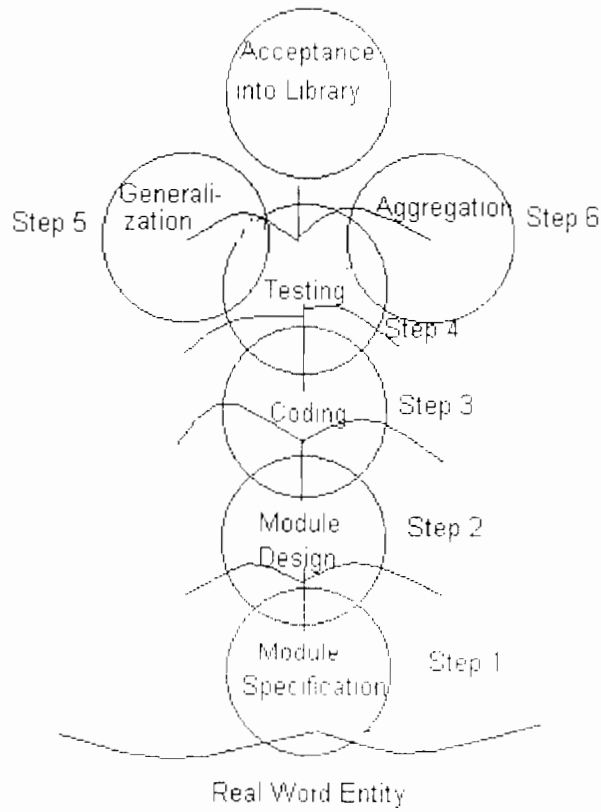

Figure 1.5 Le modèle en fontaine du cycle de vie d'un développement de module

1.2 - Survol des outils CASE

Dans le but d'aider et faciliter le processus de modélisation de façon effective, une méthode de modélisation conceptuelle devrait être automatisée dans un outil CASE. En effet, cette réalité a été reconnue pas seulement dans l'ingénierie des besoins mais aussi dans l'ingénierie de conception et ceci a conduit à une augmentation fulgurante des outils supportant différents aspects de l'ingénierie des SIs depuis une dizaine d'années.

Les outils CASE qui ont vu le jour au début des années 80s ne supportaient qu'un nombre limité de tâches à l'intérieur d'une phase de développement spécifique. Ils fournissaient essentiellement un support pour le dessin graphique des modèles et la génération de code. Par exemple, les ateliers (workbenches) tels que *Excelerator* et *Software Through Fictures* fournissaient un support de modélisation pour les modèles E/R et DFDS, tandis que les générateurs de code tel que *Install/1* et *Spectrum* se concentraient sur la génération de code pour l'application finale [McClure 89].

Les inconvénients de ces outils CASE sont doubles :

- de tels outils sont rigides (hard-coded) et inflexibles. Les développeurs devraient s'adapter à leurs façons de travailler et non l'inverse.
- de tels outils ne sont pas ouverts et sont conçus pour supporter des tâches spécifiques.

1.2.1 - Personnalisation des outils

Des tentatives d'élimination du premier inconvénient ont conduit à l'émergence des noyaux CASE (CASE shells), qui rentrent dans le cadre de méta outils CASE. Les CASE shells sont flexibles et sont conçus pour faciliter la personnalisation des outils CASE. Ils fournissent un support indépendant de la méthode en intégrant dans leur architecture un méta-modèle, et dans certains cas, une méta-méthode pour la méta-modélisation, conduisant ainsi à un outil CASE correspondant à une approche spécifique définie par l'utilisateur. Ceci est obtenu à travers le processus de méta-modélisation pendant lequel la connaissance relative à une méthode est spécifiée pour être interprétée ultérieurement. L'architecture des CASE shells permet la modification et l'extension du comportement d'un outil. On peut citer *GraphTalk* [Zerox 92], *Ramatic* [Meym 93], *Finnish Meta Edit* [Smolander 91] et le CASE Shell développé dans le projet SOCRATES [Hofstede 90] comme des exemples de CASE Shells commerciaux.

A ce propos, on distingue deux types d'utilisateurs pour les CASE Shells [Hofstede 92] : le *méta-analyste* qui définit un méta-modèle (i.e. ici, cela signifie la méta-représentation des concepts d'une méthode) décrivant la connaissance pertinente sur la méthode dans la *Métabase* (la base du méta-modèle) ; l'*analyste ou ingénieur d'information* qui pourra ensuite utiliser le CASE Shell pour supporter son propre processus de modélisation, à travers l'interprétation automatique du méta-modèle stocké dans la *Métabase* et les résultats finaux stockés automatiquement dans la *base de l'application*.

Dans le projet SOCRATES, le repository du CASE Shell contient l'information applicable au niveau méthode, i.e. la *Métabase*, qui est distinguée de l'information stockée au niveau application i.e., la base de l'application. Deux types de distinctions des connaissances au niveau méthode sont faites : la *connaissance du processus*

versus la *connaissance du produit*, et la *connaissance conceptuelle* versus la *connaissance graphique*. La connaissance du processus renferme les stratégies pour le développement de SI, tandis que la connaissance du produit définit des types de modèles, leurs composants inter-reliés et leurs règles de cohérence dans le cadre du développement des SIs. La connaissance graphique définit la notation graphique externe des concepts de modélisation stockés dans la base de la connaissance conceptuelle.
Dans *GraphTalk*, un certain nombre de contraintes et de fonctionnalités ne peuvent être définies dans le méta-modèle. Ceci est réglé par la posibilité qu'offerte *GraphTalk* de définir des procédures en langage C, via *le GraphTalk's Application Programming Interface (API)*. Ces procédures sont exécutées à la suite d'une opération effectuée par l'utilisateur. Il existe deux types de procédures : les *actions* et les *démons*. Les *actions*, exécutées à la suite d'un choix dans un menu, sont associées à un type de graphe, à un type d'objet ou à un type de liens. Les *démons* sont déclenchés à la suite d'un événement : création, suppression ou sélection d'un objet ou d'un lien, etc. Les *actions* permettent d'ajouter des fonctionnalités, tandis que les *démons* sont utilisés pour assurer la cohérence entre les graphes dans les cas où le méta-modèle ne permet de l'assurer. Tant dans une *action* que dans un *démon*, il est possible d'accéder à la base de spécifications et même de la modifier en utilisant les procédures et fonctions offertes par le langage *GQL (GraphTalk Query Language)*, qui est un langage de type *SQL*.

1.2.2 - Intégration d'outils

Le besoin de supprimer ou tout au moins d'alléger le second inconvénient passe par l'établissement d'un "pont de communication" entre les outils CASE du début des années 80s à travers une certaine forme de coopération entre eux. C'est ainsi qu'on a aboutit à la naissance d'une autre gamme d'outils CASE : ceux conçus spécialement pour l'intégration des outils CASE de la fin des années 80s. Cette catégorie d'outils CASE est souvent livrée avec un repository CASE incorporé, qui autorise le stockage de spécifications en provenance de différents composants d'outils CASE. C'est à travers le repository que le partage de spécifications par les composants d'outils CASE intégrés peut être réalisé.

1.2.2.1 - Architecture d'intégration

On trouve dans [Lindland 93] trois principales architectures d'intégration qui sont : le *cadre d'outils CASE* (*CASE Framework*), les *outils CASE intégrés* (*Integrated CASE Tools* ou *ICASE*) et l'*environnement support de projets intégrés* (*Integrated Project Support Environment* ou *IPSE*).
Le cadre d'outils CASE est une architecture ouverte d'intégration d'outils supportant différentes phases de développement. Ces outils peuvent provenir de plusieurs sources. Ils sont intégrés pour former un ensemble d'outils cohérents et uniformes qui couvre l'entièreté du cycle de vie du développement d'une application via une interface d'outil uniforme. La norme bien établie, *PCTE* (Portable Common Tools Environment)

[Boudier 88, Emeraude 92], développé dans le contexte du projet ESPRIT, *l'AD/Cycle* de IBM [Mercurro 90] et *ISO/IRDS* utilise tous une telle architecture.
L'objet des ICASEs est de supporter une approche bien définie à travers le processus de développement entier via un méta-modèle intégré. *Oracle*CASE* développé par Oracle Corp. (UK) et *Foundation* développé par Andersen Consulting [Heym 93, McClure 89] en sont des exemples typiques.
L'IPSE est un cadre qui possède beaucoup de similarités avec le cadre d'outils CASE et l'ICASE, possède des fonctionnalités additionnelles telles que des fonctions de support pour l'administration comme la gestion de configuration, le support multi-utilisateurs et le contrôle de l'accès multi-utilisateurs. *PCTE* et *ISEE* sont des exemples d'environnements qui possèdent de telles caractéristiques.
Kronlöf [Kronlöf 93] utilise une technique différente pour l'intégration d'outils. Son avis est que l'intégration de méthodes est un prérequis pour une intégration d'outils réussie. Selon sa stratégie, une nouvelle méthode devra être conçue pour un objectif donné à partir deux ou plus de méthodes existantes. L'objectif de l'intégration de méthodes est d'obtenir une méthode résultante qui combine les points forts tout en réduisant les points faibles des méthodes sélectionnées lorsqu'elles sont appliquées à un processus d'ingénierie donné.
En utilisant l'outil CAME (Computer Aided Method Engineering) décrit dans [Harmsen 93] l'ingénierie de méthode est supportée par le stockage, l'extraction et l'assemblage de fragments de méthodes. Les fragments de plusieurs méthodes de développement de SIs stockés dans la *base* dite *de méthode* peuvent être sélectionnés par des fonctionnalités de l'outil.

1.2.2.2 - L'environnement de génie logiciel basé sur PCTE

PCTE a été reconnu comme un standard de base pour la construction *des environnements de génie logiciel* (*SEEs : Software Engineering Environments*). *PCTE* [Legait 88, Thomas 89] est en fait un cadre qui fournit un support intégré pour un ensemble d'outils inter-reliés mais provenant de différents vendeurs. Ces outils coexistent à travers un ensemble de services communs. Les services pour l'intégration sont offerts à travers une technologie d'intégration incorporée. Cette technologie incorporée est supportée par le *système de gestion d'objets* (*Object Management System : OMS*) sous-jacent [Gallo 87, Minot 88].

Dans le contexte d'un SEE, l'intégration d'outils signifie la coopération de différents outils fournis par des sources différentes, par le partage d'une base de données logiquement cohérente. Par exemple, le *modèle d'information AD* de pair avec un ensemble de *services Repository* ont été construits dans l'environnement *AD/Cycle de IBM* [Mercurio 90]. Ils constituent le mécanisme de support pour l'intégration d'outils IBM et d'outils provenant d'autres vendeurs. Dans l'environnement *REBOOT* (*REuse Based on Object-Oriented Techniques*), diverses fonctionnalités sont offertes pour faciliter l'intégration d'outils et tout en même temps augmenter la productivité et la qualité dans le développement logiciel en promouvant la réutilisabilité [Faget 92].

PCTE a été accepté comme une norme (un standard) dont la principale mission est de fournir une solution en terme d'infrastructure pour intégrer ensemble plusieurs outils différents mais inter-reliés et provenant de différentes sources. Il existe plusieurs projets de recherche utilisant PCTE comme cadre, entre autres, le *projet Emeraude* dont l'objectif est de fournir une implémentation de l'interface d'outil standard pour PCTE. Cette interface, associée avec quelques outils, permettra le développement d'un IPSE. Dans PCTE, la technologie d'intégration est dérivée de l'OMS sous-jacent. Le modèle de données de OMS est basé sur le modèle E/R [Chen 76]. Il définit des objets et des relations comme entités de base de la base d'information. Dans Emeraude PCTE, une extension a été faite en intégrant le concept d'*héritage*. L'OMS agit comme un mécanisme de support pour son environnement, tout aussi bien que pour réaliser l'intégration de données issues d'outils provenant de vendeurs différents. Ainsi, les outils qui sont destinés à être installés dans son environnement utilisent le modèle de données de OMS pour réaliser l'intégration de données. Quelquefois, la base de données OMS est aussi appelée *base d'objets* (*object base*) ou *repository* de PCTE.
L'ensemble des définitions de types pour les objets, attributs et liens constituent le schéma de la mémoire d'objets PCTE. Ces définitions sont organisées en objets OMS appelés *ensembles de définition de schémas* (*Schema Definition Sets ou SDS*). Par conséquent, un SDS est un groupe de définitions de types inter-reliés.
La figure suivante (Figure 1.6) illustre l'architecture en couches des services communs d'Emeraude PCTE, les outils et la base d'objets.

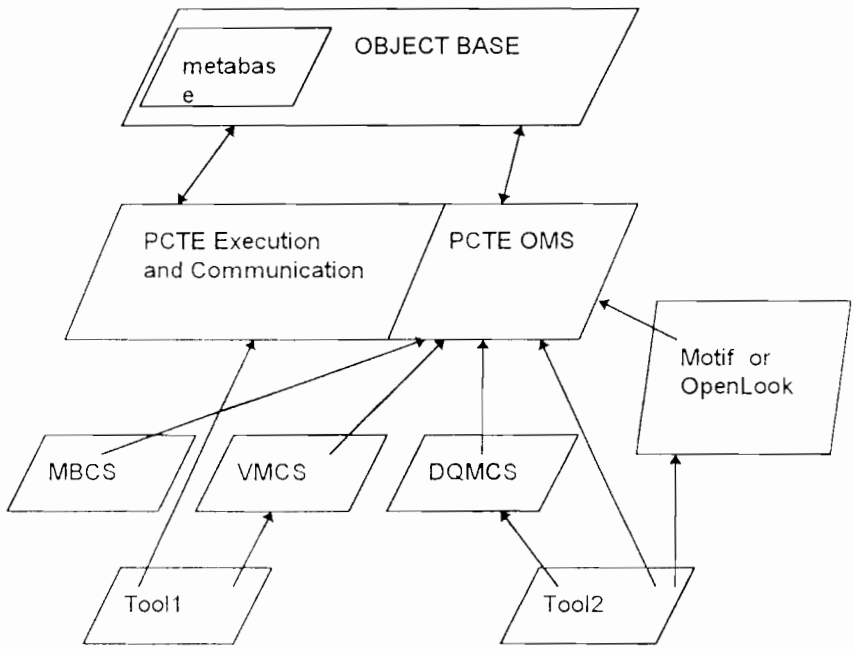

Figure 1.6 : L'architecture en couche de Emeraude PCTE

Emeraude PCTE est conçu pour les concepteurs d'environnements, les développeurs et intégrateurs d'outils dans le cadre de création de SEE. Il peut être accédé via Motif ou Open Look.

Dans la *base d'objets PCTE* sont stockés des objets, des liens et des attributs. Pris ensemble, ils forment l'information persistante de la base d'objets. Une partie de la base d'objets est appelée *Métabase*, contient les objets représentant toutes les définitions de types d'objets, de liens et d'attributs ainsi que les SDSs qui sont présents dans l'environnement de façon cohérente.

Les trois principaux services communs d'Emeraude, construits au dessus de l'OMS de PCTE sont les suivants :

- les *services communs de la Métabase* (Metabase Commun Services ou MBCS) qui peuvent être utilisés pour le stockage et l'extraction d'information sur les définitions de type d'OMS.

- les *services communs de gestion de version* (*Version Management Commun Services* ou VMCS) pour la gestion des versions d'objets.

- les *services communs de manipulation et de requête de données* (*Data Query and Manipulation Common Services ou DQMCS*) pour la requête et la manipulation de données sur la base d'objets.

Les services communs servent de support sous-jacent pour le développement d'outils.

*Chapitre 2 : la méthode O**

Le modèle conceptuel de O* se distingue par les points suivants :

-c'est un **modèle sémantique**, qui définit un ensemble cohérent de concepts permettant de représenter la totalité des aspects statiques et dynamiques d'un système d'information de telle sorte que la représentation obtenue soit proche de la manière dont l'être humain perçoit le monde réel.

-c'est un **modèle OO** , par lequel un système d'information est considéré comme une société d'objets interagissant entre eux et entretenant des relations durables. La spécification résultant de l'utilisation du modèle consiste en un ensemble de classes caractérisant à la fois la structure et le comportement de leurs objets évitant ainsi la dichotomie entre les données et les traitements qui est l'inconvénient majeur des modèles classiques.

-c'est un **modèle événementiel** : le modèle O* propose d'introduire le concept d'événement dans le paradigme OO dès l'étape de modélisation conceptuelle, afin d'autoriser la description des aspects dynamiques liés à la causalité (inférence d'événement) et liés à la concurrence (synchronisation événements ou d'opérations). La causalité est spécifiée par l'intermédiaire événements internes aux objets qui se produisent à la suite de changements d'état particuliers. La synchronisation est prise en compte dans la définition même de l'événement, qui déclenche simultanément un ensemble d'opérations qui, à leur tour, peuvent entraîner la survenance d'autres événements.

A Les concepts

2.1 Le concept d'objet et de classe

L'identification des classes résulte d'un processus de classification des objets
Une classe décrit un ensemble d'objets de même nature, qui sont qualifiés d'instance de la classe. Une classe est caractérisé par une extension et par une intention appelée schéma. L'extension d'une classe est à un instant donné, l'ensemble des objets dont elle définit la structure et le comportement. Le schéma d'une classe consiste en un ensemble d'items définissant la structure et le comportement de toutes ses instances

Exemple
"Client Diop" et "Client Diouf" peuvent être des objets potentiels. Ils sont classifiés dans la classe CLIENT

2.2 lien de composition entre une classe et un domaine de valeurs

Il associe une valeur à chaque instance d'une classe. Il est identifié par un nom qui est unique à l'intérieur d'une classe

Exemple
Âge client défini dans la classe CLIENT prend ses valeurs dans le domaine ENTIER

2.3 Le lien de composition à valeurs dans une classe

Il exprime un fort couplage de comportement entre un objet composite et un objet composant
Un lien de composition simple est représenté graphiquement par une flèche pleine allant de la classe composite vers la classe composante. Un lien de composition multiple est représenté graphiquement par une double flèche pleine allant de la classe composite vers la classe composante

Exemple COMMANDE et LIGNE DE COMMANDE

Le cycle de vie d'un objet composant LIGNE DE COMMANDE est inclus dans le cycle de vie de l'objet composite COMMANDE

classe COMMANDE
propriétés
lignes : set of(LIGNE DE COMMANDE)

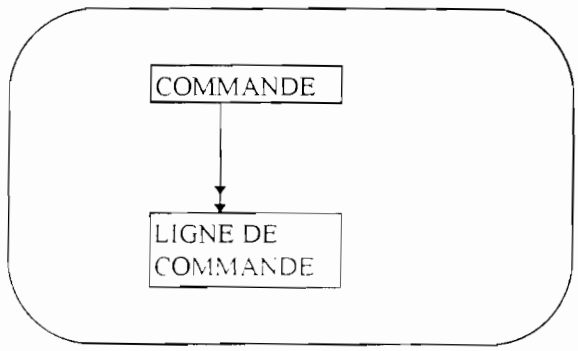

Figure 2.1 Exemple de lien de composition multiple

2.4 Le lien de référence

Il exprime un couplage faible de comportement entre un objet référençant et un objet référencé. Il permet d'établir des relations transitoires, ou temporelles entre objets par opposition aux relations structurelles représentées par les liens de composition. Un lien de référence simple est représenté graphiquement par une flèche en pointillé allant de la classe référençante vers la classe référencée
Un lien de référence multiple est représenté graphiquement par une double flèche en pointillé allant de la classe référençante vers la classe référencée.

Exemple : COMMANDE et CLIENT

classe COMMANDE
références
client : CLIENT

le cycle de vie de l'objet référençant ici COMMANDE est inclus dans le cycle de vie de l' objet référencé CLIENT

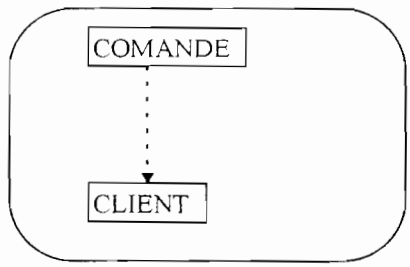

Figure 2.2 Exemple de lien de référence simple

2.5 Lien d'héritage

Un lien d'héritage entre deux classes établit une relation de spécialisation-généralisation entre chaque objet de la première classe, dit objet spécialisé et un objet de la deuxième classe, dit objet généralisé. Par extension on parlera respectivement de classe spécialisée et de classe généralisée, la classe spécialisée héritant de la classe généralisée.
La représentation de vues multiples d'un même objet est autorisé. L'objet « personne DURAND » par exemple peut être spécialisé en « étudiant DURAND » et en « salarié DURAND »
Le concept d'héritage utilisé dans le modèle est différent de celui utilisé dans la programmation OO car il est basé sur le principe de non exception. Celui ci établit que toute caractéristique spécifiée dans une classe spécialisée ne doit pas invalider les caractéristiques établies dans les classes généralisées correspondantes. Ce qui est contraire au mécanisme de surcharge utilisé en programmation OO .

Le lien d'héritage est représenté graphiquement par une double flèche allant de la classe spécialisée vers la classe généralisée correspondante

Exemple
classe PERSONNE

classe ÉTUDIANT
hérite de PERSONNE

classe SALARIE
    hérite de PERSONNE

classe ETUDIANT_SALARIE
    hérite de ÉTUDIANT, SALARIE

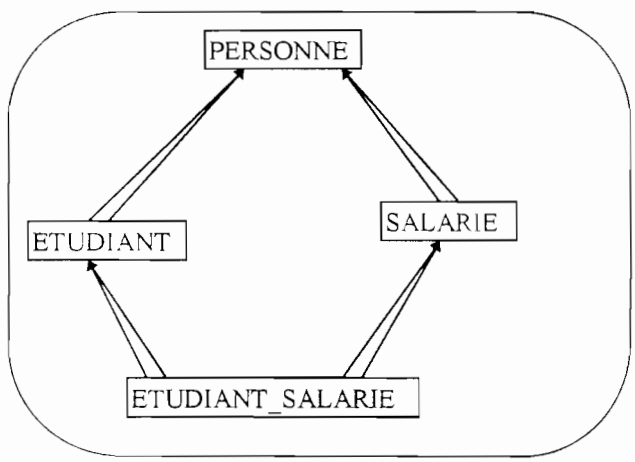

Figure 2.3 Exemples de liens d'héritage

2.6 Contraintes statiques

2.6.1 Contraintes d'attribut

Les contraintes d'attribut permettent d'exprimer des contraintes sur les valeurs que peuvent prendre les attributs ainsi que sur les cardinalités des attributs de type collection, c'est à dire des attributs multiples (correspondant à des liens de composition ou de référence multiple).
Une contrainte d'attribut peut s'écrire sous la forme d'une assertion booléenne constituée d'un ensemble de prédicats combinés par les opérateurs booléens usuels

Exemples
classe SALARIE
 propriétés
   nom : CHAINE
   âge : ENTIER
   salaire courant : REEL
   salaire précédent : REEL
 assertions
   âge >= 18
   (salaire courant > salaire précédent) ou (âge <= 25)

classe VOITURE
propriétés
roues : ensemble (ROUE)
assertions
card(roues) = 4

2.6.2 Contraintes d'unicité

Une contrainte d'unicité permet d'établir l'unicité des valeurs prises par un ou un ensemble d'attributs d'une classe
Exemple
classe FACTURE
propriétés
montant total : REEL
références
commande : COMMANDE
assertions
unique (commande)

2.6.3 Contraintes d'héritage

Une contrainte d'héritage a pour fonction de restreindre les possibilités d'existence d'objets spécialisés, pour chaque objet d'une classe généralisée. Trois types de contraintes d'héritage peuvent être établies:

-la disjonction :
à tout objet de la classe généralisée ne peut correspondre au plus qu'un seul objet spécialisé dans une de ces classes

Exemple : la spécialisation de VEHICULE en VOITURE et CAMION ; un véhicule peut être une voiture ou un camion mais pas les deux, et peut aussi être ni une voiture ni un camion.

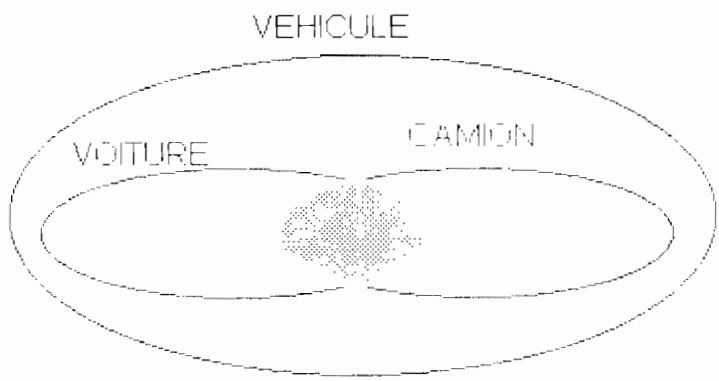

Figure 2.4 Exemple de contrainte de disjonction

-la couverture:
à tout objet de la classe généralisée doit nécessairement correspondre au moins un objet spécialisé appartenant à une de ces classes

Exemple: la spécialisation de **PERSONNE** en **ETUDIANT** et **ENSEIGNANT** une personne peut être un étudiant, un enseignant ou les deux à la fois, et doit être nécessairement un étudiant ou un enseignant

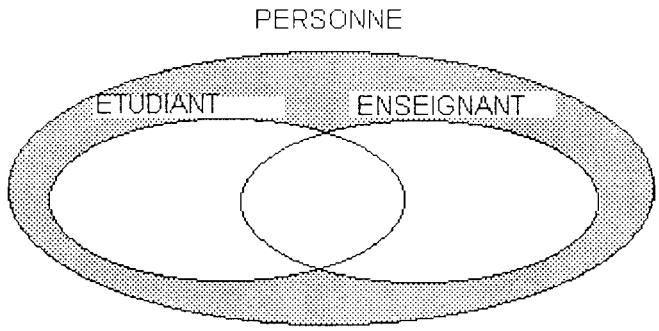

Figure 2.5 Exemple de contrainte de couverture

-la partition:
exprime à la fois une contrainte de disjonction et une contrainte de couverture : à tout objet de la classe généralisée doit nécessairement correspondre exactement un objet spécialisé appartenant à une des classes présentes dans la contrainte

Exemple la spécialisation de PERSONNE en HOMME et FEMME une personne est soit un homme soit une femme, ne peut être les deux à la fois et ne peut être autre chose.

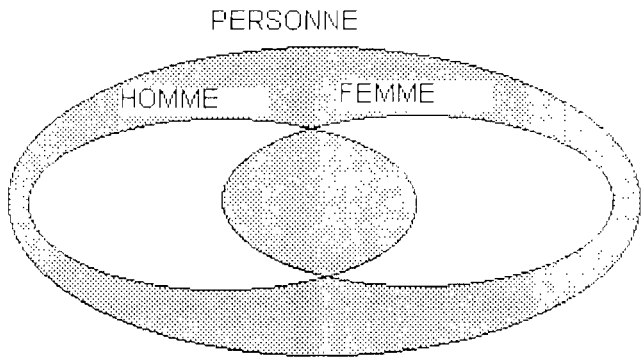

Figure 2.6 Exemple de contrainte de partition

2.7 L'opération

Une opération est une procédure dont le déclenchement effectue la création, la modification des valeurs ou la suppression d'un objet d'une classe.

Exemple
classe COMMANDE
propriétés
lignes : set of (LIGNE DE COMMANDE)
date de création : DATE
date de livraison : DATE
date de facture : DATE
références
client : CLIENT
opérations
créer
livrer
facturer
annuler

2.8 L'événement

Une occurrence d'événement est une projection d'un phénomène de la réalité organisationnelle, qui se produit en un point du temps et a un impact sur le système d'information.
Un **événement** décrit un ensemble d'occurrences événement de même nature. Cette description comprend la cause de la survenance d'une occurrence événement et son impact sur les objets du système d'information.

On peut caractériser un événement selon sa condition d'occurrence :
. un événement interne constate le changement d'état remarquable d'un objet du système d'information
-un événement temporel survient à un instant déterminé à l'avance
-un événement externe a pour origine un stimulus en provenance de l'environnement du système d'information, il est la plupart du temps associé à un message.

Les syntaxe des événements internes est
<événement interne ::= <nom événement>
prédicat
<prédicat interne>
déclenchement
[<déclenchement d'une opération>]]+

La syntaxe des événements temporels est
<événement temporel ::= <nom événement>
prédicat
<prédicat temporel>
déclenchement
[<déclenchement d'une opération>]]+

Les syntaxe des événements externes est
<événement externe ::= <nom événement>
[message
:<message>]
déclenchement
[<déclenchement d'une opération>]+
<message> ::= (<paramètre>,[, paramètre]*)

Quelque soit le type d'un événement, la spécification des déclenchements d'opérations est effectuée de façon identique:
<déclenchement d'une opération >::=<nom d'opération> **sur** <nom de classe >
[**pour** <facteur>]
[**si** <condition>]

<facteur> ::= <condition de facteur>, <message>

Exemples:
"Arrivée de la commande d'un Macintosh LC par Diouf le 3 Juillet 1995"
"Facturation de la commande numéro 2355"

classe ENVIRONNEMENT
événements
arrivée d'une commande
message
structure (COMMANDE)
déclenchement
créer **sur** COMMANDE
créer **sur** CLIENT **si** le client n'existe pas

2.9 Les contraintes dynamiques

Les contraintes dynamiques ont pour but de modéliser les aspects proscriptifs de la dynamique d'un système d'information Les contraintes dynamiques sont spécifiées par l'intermédiaire de graphes de transitions d'état.

2.9.1 la classe d'état

L'état d'un objet donné à un instant donné est défini comme étant l'ensemble de ses valeurs.Une classe d'états est un sous-ensemble des états possibles des objets d'une classe, pour lesquels les objets présentent un comportement spécifique.

Exemple
Considérons par exemple les objets d'une classe LIGNE DE COMMANDE. L'état caractérisant une ligne de commande est un triplet de valeurs constituée de :
-une quantité commandée, appartenant au domaine ENTIER
-une identité de produit appartenant à la classe PRODUIT
-un statut : créée, livrée ou facturée.

Seule la troisième valeur influe sur le comportement d'une ligne de commande : suivant son statut, les événements qui peuvent l'affecter sont différents. Cela conduit à créer les classes d'états CRÉÉE, LIVRÉE et FACTURÉE

2.9.2 la transition d'état

Le concept de transition d'état répond à deux objectifs :
-déterminer l'ensemble des événements applicables à un objet se trouvant dans une classe d'états donnée
-déterminer la (ou les) classe(s) d'états résultant de l'impact d'une occurrence événement sur un objet.

Une transition d'état est un triplet (*classe d'états initiale, opération, classe d'états finale*), où :
- classe d'états initiale est une classe d'états possible de l'objet avant exécution de opération
- classe d'états finale est une classe d'états possible de l'objet après exécution de opération.
Classe d'états initiale et classe d'états finale sont soit exclusives, soit identiques

Exemple de l'objet ligne de commande
La transition d'état (LIVRÉE, facturer, FACTURÉE) stipule qu'il est possible d'exécuter l'opération facturer quand cet objet est dans la classe d'états LIVRÉE, et d'autre part que cet objet peut passer dans la classe d'états FACTURÉE

2.9.3 Le graphe de transitions d'états
Le concept de graphe de transitions d'état permet, à travers la spécification exhaustive des transitions d'état, d'exprimer les contraintes dynamiques des objets d'une classe.

Un graphe de transition d'état est un graphe dont les noeuds sont des classes d'état formant une partition sur les états possibles des objets d'une classe et dont les arcs sont des transitions d'état possibles entre ces classes d'états. La classe de non existence S0 est ajoutée à la partition .

Exemple :

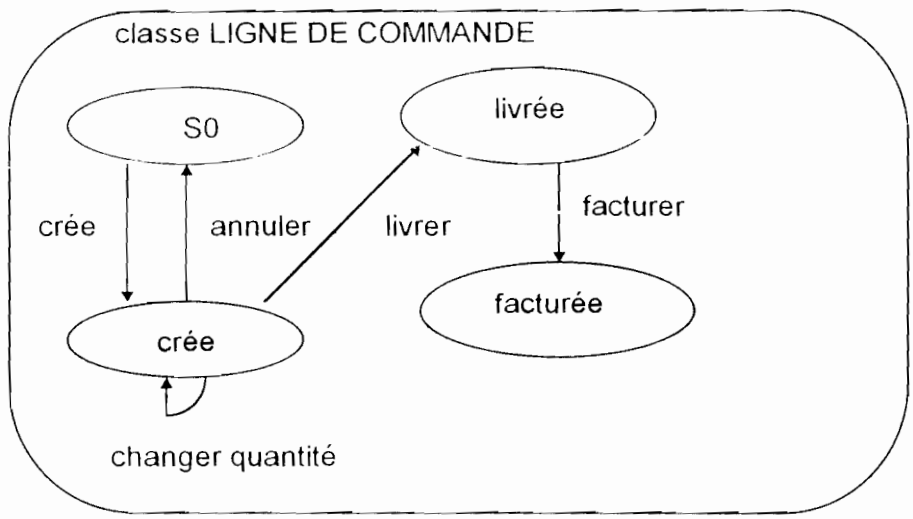

Figure 2.7 le graphe de transitions d'état d'une ligne de commande

B Le processus de modélisation

Il s'agit du processus d'aide de l'analyste dans sa démarche de modélisation. Il est constitué d'heuristiques et de règles méthodologiques regroupées en six activités ou étapes.
Chaque étape correspond à un point de vue particulier que l'analyste doit adopter au cours de sa démarche.
Les grandes lignes que doit suivre l'analyste pour atteindre la spécification complète d'un système d'information sont les suivantes :

Étape 1 : Inventaire des objets du monde réel
Cette étape initialise le processus de modélisation. Elle consiste à étudier le domaine d'application dans le but d'isoler des objets potentiels.

Étape 2 Description initiale des objets
Le domaine d'application est étudié dans le but d'identifier les caractéristiques pertinentes des objets obtenus à l'étape précédente. La classification des objets de même nature conduit à identifier un premier ensemble de classes.

Étape 3 Étude locale des classes
On décrit à cette étape, au moyen des concepts du modèle , les caractéristiques statiques et dynamiques des objets d'une classe, chaque classe étant considérée

indépendamment des autres. Les concepts d'attribut et d'assertion concernent les aspects statiques. Les concepts d'opération et de graphes de transition d'états concernent les aspects dynamiques.

Étape 4 Étude globale des aspects statiques
Cette étape inclut l'étude des liens structurels entre classes. Cette étude se base sur le graphe statique qui visualise les liens de composition, de référence et d'héritage.

Étape 5 Étude globale des aspects dynamiques
Elle prend en compte l'ensemble des classes et se concentre sur les interactions de comportement entre classes :
-exécution complète des opérations déclenchées par chaque événement
-exécution complète des événements déclenchant chaque opération
-description des conditions et des facteurs.
Cette étude se base sur le graphe dynamique qui visualise les événements et leurs relations avec les opérations qu'ils déclenchent.

Étape 6 Validation et Vérification
L'objectif de cette étape est de procéder à la validation et à la vérification du schéma conceptuel obtenu. La validation est réalisée au moyen de contrôles de cohérence et de complétude. La vérification consiste en l'étude critique du schéma conceptuel par les utilisateurs finaux de l'application.

Le modèle en fontaine sur lequel est basé le processus, n'impose pas qu'une étape doive être complétée avant de passer à l'étape suivante. Il existe au contraire une importante itération entre les étapes du processus de modélisation.

Chapitre 3 : l'éliciteur

3.1 le modèle des cas d'utilisation

Plusieurs approches basées sur les scénarios existent, un cadre pour les classifier a été proposé dans [Rolland97] et J.M.Caroll soutient dans [Caroll96] que la perspective scénario dépasse les premières phases de développement à savoir l'analyse des besoins et la conception et peut jouer un rôle de guide à travers tout le cycle de développement d'un système d'information. L'approche que nous avons suivi est de [Regnell 95] parce qu'elle fournit une représentation structurée et semi-formelle des cas d'utilisation.

3.1.1 Intérêts de la modélisation des cas d'utilisation

Lorsque l'on traite des systèmes complexes un passage trop rapide en une seule étape de la description informelle des besoins fournie par le client vers une spécification formelle des besoins peut avoir plusieurs conséquences négatives, tel un fossé sémantique important entre la description des besoins et la spécification des besoins ou l'incomplétude de cette dernière.
Il est aussi très difficile de produire une spécification formelle sans une compréhension profonde de ce que le client et les utilisateurs finaux espèrent du système et de comment ils ont l'intention de l'employer en pratique. Ce type d'information est rarement disponible au début du développement du système

3.1.2 l'ingénierie des besoins fondée sur l'utilisation [Regnell 95]

Nous travaillons avec le concept de cas d'utilisation introduit dans [Jacobson 92] en tenant compte de l'amélioration et de la formalisation apportées par [Regnell 95].
Le **modèle des cas d'utilisation** spécifie la fonctionnalité que le système doit offrir du point de vue de l'utilisateur [Jacobson 92]. Ce modèle utilise des ***acteurs*** pour représenter les rôles que l'utilisateur peut jouer, et des ***cas d'utilisation*** pour représenter ce que les utilisateurs devraient être capables de faire avec le système. Chaque cas d'utilisation est ***une suite complète événements*** au sein du système, vu du point de vue de l'utilisateur.
Dans [Regnell 95] il est proposé un processus d'ingénierie des besoins orienté utilisation UORE (Usage Oriented Requirements Engineering),visant à améliorer

l'original UCDA (Use Case Driven Analysis) de la méthode Objectory [Jacobson 92]. UORE est constitué de deux phases : la ***phase d'analyse*** et la ***phase de synthèse***. La phase d'analyse a comme entrée une description informelle des besoins et produit le ***modèle des cas d'utilisation*** contenant la description des acteurs et des cas d'utilisation. Ce modèle à son tour est utilisé comme entrée à la phase de synthèse qui formalise les cas d'utilisation, les intègre et crée le ***modèle d'utilisation synthétisé***. (Figure 3.1)

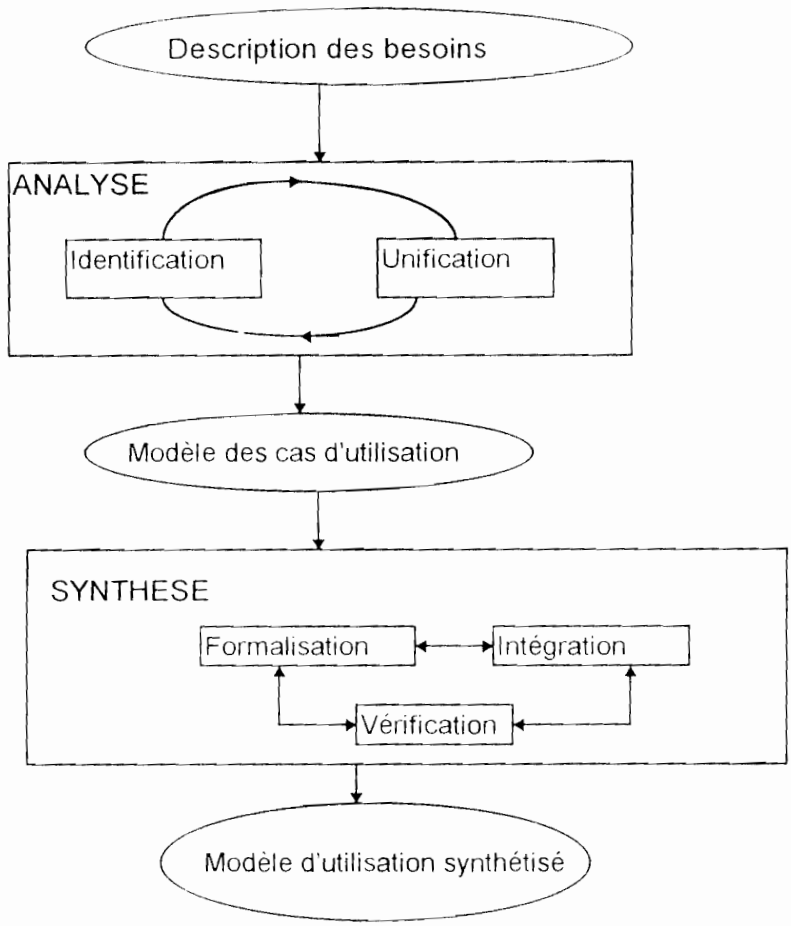

Figure 3.1 : le processus UORE

Le modèle d'utilisation synthétisé est un graphe orienté avec trois types de noeuds : les actions de l'utilisateur, les actions du système, les contextes d'invocation et de terminaison.

En résumé, les principales contributions de UORE par rapport à l'UCDA sont :

--l'amélioration du concept d'acteur et de cas d'utilisation

--l'application stricte du point de vue acteur unique pour chaque cas d'utilisation

--la formalisation des descriptions des cas d'utilisation
--l'idée de synthèse des cas d'utilisation
--le modèle d'utilisation synthétisé
--le processus de l'ingénierie des besoins orientés utilisation

Le modèle d'utilisation synthétisé est destiné à constituer une partie de la spécification des besoins et un modèle de référence pour la validation et la vérification. Il capture les aspects interreliés suivants :

-catégories des utilisateurs du système et leurs objectifs,
-les objets du domaine, leurs attributs et leurs opérations,
-les stimulis et réponses de la communication utilisateur - système,
-les actions du système et de l'utilisateur, leurs combinaisons possibles et contextes d'utilisation,
-les scénarios de l'utilisation du système, leurs flots événements, et conditions de déclenchement.

3.2- l'exemple du guichet automatique de banque

Pour illustrer UORE nous utiliserons un exemple bien connu celui du guichet automatique de banque(GAB). Le GAB offre fondamentalement 2 services : le retrait d'espèces et le contrôle de compte
Nous présentons ci-dessous une partie du résultat de la partie analyse de la description informelle du GAB:

Liste des acteurs

l'acteur « client du GAB » : le client du GAB utilise le GAB pour retirer des espèces ou contrôler le solde de son compte
le superviseur du GAB : il supervise et maintient le fonctionnement du GAB
la base de données du GAB: le système externe qui maintient l'information sur le compte

Quelques cas d'utilisation pour l'acteur Client GAB

I Retrait d'espèces, Cas normal

Acteur client du GAB

1.I Condition d'invocation
1.IC 1 le système est prêt pour les transactions
1. CR Conditions à remplir(Flow Condition)

1.FC.1 la carte de l'utilisateur est valide
1.FC.2 l'utilisateur entre un code valide
1.FC.3 l'utilisateur entre un montant valide
1.FC.4 la machine dispose du montant en espèces
1.FE. Flot des événements
1.FE.1 l'utilisateur insère la carte
1FE.2 le système vérifie que la carte est valide
1.FE.3 Une invite pour entrer le code est affiché
1.FE4 l'utilisateur entre le code
1.FE.5 le système teste que le code est valide
1.FE.6 une invite « Entrer le montant ou sélectionner solde » est affichée
1.FE.7 l'utilisateur tape le montant
1.FE.8 le système teste que le montant est valide
1.FE.9 le système collecte les espèces
1.FE.10 les espèces sont éjectées
1.FE.11 une invite « prenez les espèces » est affichée
1.FE.12 l'utilisateur prend les espèces
1.FE.13 la carte est éjectée
1.FE.14 une invite « retirez la carte « est affichée
1.FE.15 l'utilisateur prend la carte
1.FE.16 le système collecte l'information pour le reçu
1.FE.17 le reçu est imprimé
1.FE.18 une invite « prenez le reçu » est affichée
1.FE.19 l'utilisateur prend le reçu
1.TC Condition de terminaison
1.TC.1 le système est prêt pour les transactions

2 Retrait d'espèces, montant non valide

Acteur client du GAB

2 IC Conditions d'invocation
2.IC.1 identique à 1.IC.1
2. FC Conditions à remplir(Flow Condition)
2.FC.1 les mêmes de 1.FC.1 jusqu'à 1.FC.2
2.FC.2 l'utilisateur tape un montant non valide
2.FE Flot des événements
2.FE.1 les mêmes qu'en 1.FE.1 jusqu'à 1.FE.8

2.FE.2 le message « Montant invalide » est affichée

2.FE.3. une invite « réessayez » est affichée

2.FE.4 le système avorte la transaction

2.FE.5 identique qu'en 1.FE.13 jusqu'à 1.FE.15

2 TC Condition de terminaison

2.TC.1 identique à 1.TC.1

3 Contrôle du compte, cas normal

Acteur client du GAB

3.IC. Conditions d'invocation

3.IC.1 identique qu'en 1.IC.1

3. FC Conditions à remplir(Flow Condition)

3.FC.1 identique qu'en 1.FC.1 jusqu'à 1.FC.2

3.FE Flot des événements

3.FE.1 identique qu'en 1.FE.1 jusqu'à 1.FE.6

3.FE.2 l'utilisateur sélectionne l'item solde

3.FE.3 le système collecte l'information concernant le solde

3.FE.4 le solde est affiché

3.FE.5 identique à 1.FE.13 jusqu'à 1.FE.19

3.TC Condition de terminaison

3.TC.1 identique que à 1.TC.1

Un exemple de graphe d'utilisation synthétisé est donné dans la Figure 3.2. Ce graphe d'utilisation synthétisé est une intégration de plus de cas d'utilisation que ceux qui sont présentés là haut :

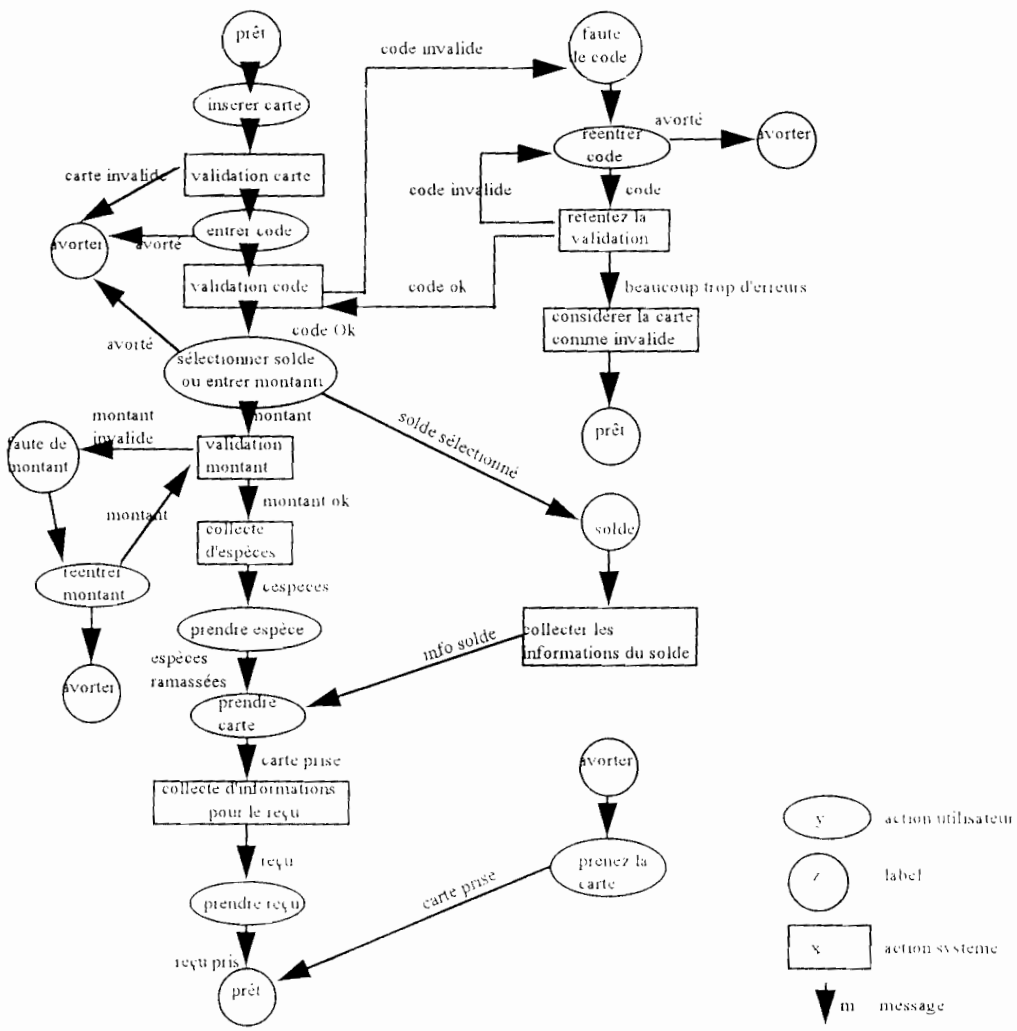

Figure 3.2 : Le graphe d'utilisation synthétisé du GAB pour l'acteur Client GAB

3.3 les règles de passage du formalisme UORE au formalisme O*

Notre contribution [Baldé 96] a été grâce à la puissance de la représentation sémantique de O* [Brunet 93] de transformer simplement un modèle d'utilisation synthétisé en deux graphes dynamiques spécifiés respectivement sur les classes O* ACTEUR et SYSTÈME.
Nous énonçons les règles de passage suivantes :
Pour chaque graphe d'utilisation synthétisé nous spécifions deux classes O*:
-une classe dérivée de la classe O* ACTEUR représentant l'acteur du graphe d'utilisation synthétisé et
-une classe SYSTÈME encapsulant les objets d'analyse intervenant dans le déroulement du graphe d'utilisation synthétisé

Les règles de passage du graphe d'utilisation synthétisé vers les graphes dynamiques bruts sont les suivantes :
-à chaque action système d'un graphe d'utilisation synthétisé nous associons une opération O* spécifiée sur la classe SYSTÈME correspondante
-à chaque action utilisateur d'un graphe d'utilisation synthétisé nous associons une opération O* spécifiée sur la classe ACTEUR correspondante

-à chaque action utilisateur, nous associons un événement externe O* correspondant à la fin de l'action utilisateur ; cet événement est spécifié à l'intérieur de la classe acteur correspondante ; le message O* de événement externe correspond aux libellés des « messages UORE » du graphe d'utilisation synthétisé ; les opérations déclenchées sont les opérations associées aux actions cibles des messages UORE provenant de l'action utilisateur ; les conditions O* de déclenchement respectives des opérations correspondent aux conditions UORE d'emprunt du chemin du message UORE dans le graphe d'utilisation synthétisé

-à chaque action système, nous faisons correspondre un événement interne correspondant à la fin de l'action système ; cet événement est spécifié sur la classe SYSTÈME ; le prédicat O* de événement interne est systématiquement vérifié ; les opérations déclenchées sont les opérations associées aux actions cibles des messages UORE provenant de l'action utilisateur ; les conditions O* de déclenchement respectives des opérations correspondent aux conditions UORE d'emprunt du chemin du message UORE dans le modèle d'utilisation synthétisé .

Nous présentons ci-dessous (Figure 3.3) une illustration de cette mise en correspondance en prenant le graphe d'utilisation synthétisé d'un client d'un guichet automatique de banque (GAB) présenté là haut .

Figure 3.3: Classes O* de description d'un cas d'utilisation classique (GAB).

classe Client GAB ***hérite de*** ACTEUR
opérations
InsérerCarte
EntrerCode
EntrerMontantouSelectSolde
PrendreEspèces
PrendreCarte
PrendreReçu
ReentrerMontant
RéentrerCode

évènements
carte Insérée
message
(numéro de compte)
déclenche
ValidationCarte ***sur*** SYSTÈME
finEntrerCode
message
(Code)
déclenche
ValidationCode ***sur*** SYSTÈME
si CodeEntré
PrendreCarte ***sur*** Client GAB
si Avorté
FinEntrerMontantouSelectSolde
message
( )
déclenche
ValidationMontant ***sur*** SYSTÈME
si MontantEntré
PrendreCarte ***sur*** Client GAB
si Avorté
CollecteInfosSolde ***sur*** SYSTÈME
si solde sélectionné

finRéentrerCode
message
(Code)
déclenche
RetenterValidation *sur* SYSTÈME
si CodeEntré
PrendreCarte *sur* Client GAB
si Avorté

finRéEntrerMontant
message
(Montant)
déclenche
ValidationMontant *sur* SYSTÈME
si MontantEntré
PrendreCarte *sur* Client GAB
si Avorté

finPrendreEspèces
message
()
déclenche
PrendreCarte *sur* Client GAB

finPrendreCarte
message
()
déclenche
CollecteInfoReçu *sur* SYSTÈME
MettreSystèmePretPourTransaction *sur* SYSTÈME
si suiteàTransactionAvorté

classe SYSTÈME
opérations
ValidationCarte
ValidationCode
ValidationMontant
CollecteEspèces
CollecteInfoReçu
RetenterValidation
RejeterCarteInvalide
CollecteInfosSolde
MettreSystèmePretPourTransaction
événements
FinValidationCarte
prédicat
(TRUE)
déclenche

EntrerCode *sur* Client GAB
si CarteOk
PrendreCarte *sur* Client GAB
si avorte

FinValidationCode
prédicat
(TRUE)
déclenche
EntrerMontantouSelectSolde *sur* Client GAB
si CodeOk
RéentrerCode *sur* Client GAB
si CodeInvalide

FinValidationMontant
prédicat
(TRUE)
déclenche
CollecteEspèces *sur* SYSTÈME
si MontantOk
RéentrerMontant *sur* Client GAB
si MontantInvalide

FinRetenterValidation
prédicat
(TRUE)
déclenche
RejeterCarteInvalide *sur* SYSTÈME
si TropErreur
RéentrerCode *sur* Client GAB
si CodeInvalide
EntrerMessageOuSelectCode *sur* Client GAB
si CodeOk

FinCollecteEspèces
prédicat
(TRUE)
déclenche
PrendreEspèces *sur* Client GAB

FinCollecteInfoSolde
prédicat
(TRUE)
déclenche
PrendreCarte *sur* Client GAB

FinCollecteReçu
prédicat
(TRUE)
déclenche
PrendreReçu *sur* Client GAB

3.4 Objectifs de réalisation

Cette partie de l'éliciteur que nous nous proposons de construire permettra :
-d'éditer les modèles d'utilisation synthétisés,
-de contrôler que chaque scénario d'utilisation abstrait est un chemin possible dans le graphe d'utilisation synthétisé correspondant,
-de découvrir de nouveaux scénarios d'utilisation grâce à un parcours systématique des modèles d'utilisation synthétisé,
-de traduire automatiquement un modèle d'utilisation synthétisé en graphes dynamiques O* que l'analyste pourra ensuite exécuter grâce au prototypeur puisqu'il dispose de spécifications O*, donnant ainsi la possibilité de générer des prototypes de l'interface utilisateur.

Chapitre 4 : le gestionnaire de la persistance

La persistance des méta-données et des données de l'application finale est obtenue actuellement à l'aide d'un fichier. Ce fichier est chargé intégralement en mémoire au début de chaque session. Cette solution peut être représenté par le schéma suivant (Figure 4.1) :

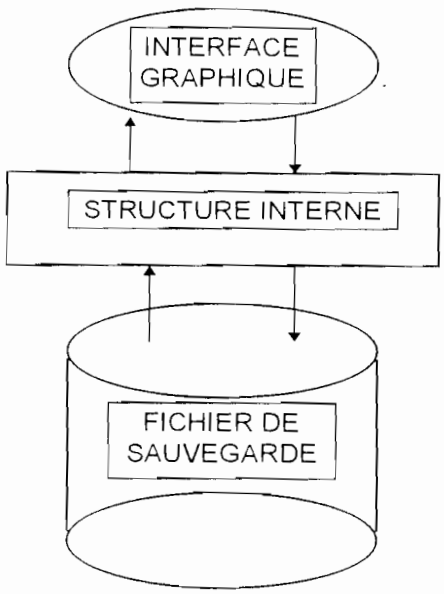

Figure 4.1 la persistance du prototype à l'état actuel

Notre objectif consiste précisément à étudier la possibilité d'assurer la sauvegarde et le chargement automatique par un système de gestion de base de donnée (SGBD) en tables relationnelles sur le disque dur. Nous en avons réalisé une étude de faisabilité [Arial D / 95-3]. Cela se traduit par le schéma suivant (Figure 4.2) :

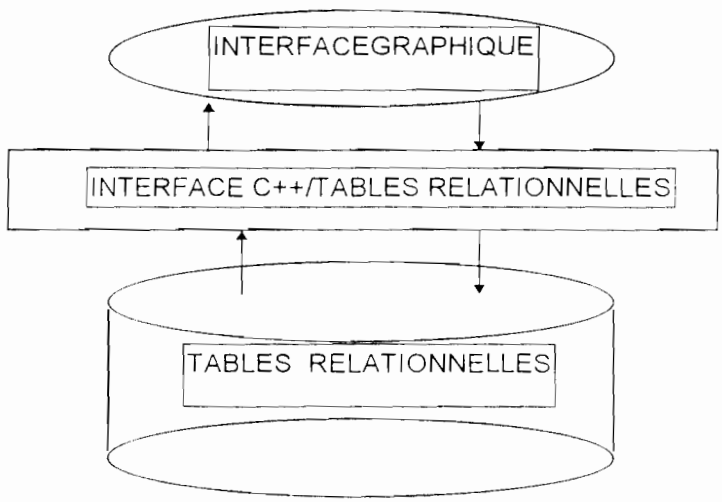

Figure 4.2 la persistance du prototype à l'état final

De ce fait notre démarche peut être décomposée en 4 étapes :
- la transformation d'un modèle O* en modèle relationnel ;
- la définition d'une interface permettant de simuler une base de données OO à l'aide d'une base de données relationnelle Paradox. Nous avons choisi Paradox puisqu'il existe un package de Borland (Paradox Engine 2.0) permettant de gérer des bases de données sous la forme de tables relationnelles au standard Paradox à l'aide des langages C et C++ [Mollaret 92].
- l'implémentation des interfaces permettant de réaliser ces fonctionnalités.
- la gestion éventuelle d'un swapping pour le chargement des tables dont on a besoin.

4.1-les règles de transformation du modèle O* en modèle relationnel

4.1.1 - Principes de transformation

L'objectif est de transformer les classes O* en tables relationnelles et de traduire les autres concepts O* dans le même formalisme. Ainsi les règles que nous avons adoptées pour faire la correspondance entre un modèle O* vers un modèle relationnel sont les suivantes :

-Règle 1 : Cas d'un lien statique d'attribut

Soit une classe O* C.
Soit P une propriété reliée à C par un lien statique d'attribut.
Alors la table relationnelle issue de la transformation de la classe C contient l'attribut P.

Exemple

classe CLIENT
propriétés
Code : ENTIER;
nom : STRING;
Adresse : STRING;

la transformation donne :

CLIENT = {Code, Nom, Adresse}

- Règle 2 : cas d'un lien de composition à valeur dans une classe

C'est l'équivalent d'une relation de type père-fils.
Dans ce cas l'identifiant de la table associée à la classe origine du lien est dupliqué dans la table associée à la classe extrémité du lien.

Exemple

composition multiple

classe COMMANDES
propriétés
numcommande : set of (LIGNE_COMMANDE)

classe LIGNE_COMMANDE
propriétés
désignation : STRING
quantité : ENTIER
prix : REEL

La transformation donne

LIGNE_COMMANDE = {numcommande, désignation, quantité, prix}

composition simple

classe CLIENT
propriétés
code : ENTIER
nom : STRING
adresse : STRING
compte : COMPTE

classe COMPTE
propriétés
numcompte : ENTIER
solde : ENTIER

La transformation donne
COMPTE = {code,numcompte, solde}

Règle 3 : Cas d'un lien de référence

- simple

l'identifiant de la table associée à la classe extrémité du lien est dupliqué dans la table associée à la classe origine du lien

Exemple

classe COMMANDES
propriétés
numcommande : ENTIER
références
principal : CLIENT

La transformation donne

COMMANDES = {numcommande, code}

- multiple

> Il s'agit de créer une troisième table avec comme identifiant les identifiants des tables associées à la classe origine et la classe extrémité du lien

Exemple

classe VÉHICULE
propriétés
numvehicule : ENTIER
références
options : set of (OPTIONS)

classe OPTIONS
propriétés
code_option : ENTIER

La transformation donne

VEHICULE_OPTIONS = { numvehicule, code_option }

- Règle 4 : Cas d'un lien d'héritage

L'héritage, du fait de sa complexité nous a amené à considérer différentes solutions au niveau de la transformation :
En premier lieu nous avons pensé adopter une démarche ascendante. Cette démarche consiste à dupliquer dans les tables associées aux classes de plus bas niveau (classes

spécialisées), les propriétés de l'ensemble des classes de niveau plus élevé (classes généralisées), puis remonter d'un niveau et appliquer le même principe.
Cependant cette approche ne permet pas de représenter les contraintes de couverture de disjonction et de partition. Cette solution pourrait entraîner une redondance des propriétés héritées.

La deuxième option consiste à reporter l'identifiant de la classe généralisée dans les classes spécialisées (l'identifiant seulement). Cette option a l'avantage de minimiser les redondances des propriétés dans les différentes tables correspondant aux classes spécialisées. Cependant le cas des classes généralisées abstraites qui n'ont pas de propriétés nous conduit à énoncer une troisième solution. C'est cette dernière qui sera retenue pour la suite.

Cette solution vient en complément de la deuxième et va consister à traduire le fait qu'une classe spécialisée hérite de tous les liens de sa classe généralisée. Cette remarque n'a d'intérêt que si la classe généralisée est abstraite.

Exemple

classe CLIENT **hérite de** PERSONNE
propriétés
code : ENTIER
compte : COMPTE
opérations
créer
modifier
supprimer

classe PERSONNE
propriétés
matricule : ENTIER
nom : STRING
adresse : STRING
opérations
créer
modifier adresse
supprimer

La transformation donne

CLIENT = { code, matricule }
PERSONNE = { matricule, nom, adresse }

COMPTE = { numcompte, solde, code }

Gestion des identifiants
A la suite de l'application des règles, si une classe a une assertion avec la clause unique, l'attribut figurant dans cette clause est considéré comme identifiant. Dans le cas contraire, il faut ajouter dans la table correspondante un attribut numéro qui va alors servir d'identifiant.

Partition - Couverture - disjonction
L'application des règles ainsi définies ne semble pas gérer les concepts de partition, de couverture et de disjonction. Cet aspect sera pris en charge au niveau de l'implémentation du système.

Cas de la partition
Pour les classes spécialisées, l'intersection des projections sur l'identifiant de deux classes quelconques doit être vide tandis que l'union sera égale à la classe généralisée.

Cas de la disjonction
L'intersection des projections sur l'identifiant des classes spécialisées doit être vide et l'union strictement incluse dans la classe généralisée.

Cas de la couverture
La réunion de toutes les projections des classes spécialisées doit être égale à la classe généralisée.

La gestion de ces cas sera réalisée par des procédures de pré-insertion et de pré-suppression qui seront définies au niveau de l'implémentation du système.
Ces procédures vont effectuer les contrôles nécessaires sur les instances des tables participant à un lien d'héritage suivant les trois types de contraintes ainsi indiquées.

Règle d'optimisation

R1 :

Si une table représente une classe généralisée qui ne comporte pas de lien statique d'attribut, alors elle est supprimée.

R2 :
Elle vise à réorganiser les tables issues d'une même arborescence suivant les règles d'héritage.

Soient G une classe généralisée, et A l'ensemble de ses attributs.
il existe alors n > 0 et ($S_i = A_i$) 1<=i<=n, S_i étant une classe spécialisée de G avec A_i comme ensemble d'attributs.
S'il existe k tel que $\forall$ i $\in$ {1..n} $A_i \subseteq A_k$
alors construire G' telle que
$G' = A \cup (A_k - A) \cup$ {Type} avec type ayant comme domaine { S_i } 1 <= i <= n.
Il suffit ainsi d'éliminer les S_i du schéma et de remplacer G par G'.

NB :
Cette règle ne s'applique que si la relation d'héritage est une disjonction ou une partition. S'il s'agit d'une partition : l'attribut type est déclaré comme **not null**. Par contre, le fait que l'attribut type puisse prendre une valeur nulle permet de représenter la disjonction.

4.1.2 Application des règles au métamodele O*

a) Méthode

Si une classe a des liens statiques d'attribut, créer la table ; Pour chaque classe, identifier les liens de référence simples et/ou les liens d'héritage qui ont cette classe comme origine et les liens de composition qui ont cette classe comme extrémité faire pour chaque cas le traitement approprié. Le métamodèle de la méthode O* se trouve en annexe 1.
Ceci nous donne alors un premier schéma relationnel du métamodèle ; ce premier schéma sera proposé au concepteur qui pourra le valider ou l'optimiser suivant les règles d'optimisation proposées. Le choix d'automatisation de l'optimisation pourrait être envisagé.

b) Premier Schéma Relationnel

CLASSE = {nom de classe}
GTE = {nom de GTE, nom de classe}
PROPRIÉTÉ = {nom de propriété, nom de domaine, nom de classe}
OPÉRATION = {nom d'opération, nom de classe, spécification d'opération}
COMPOS DOMAINE = {nom de domaine, libelle domaine agrégat,libellé composant domaine agrégat}
DOMAINE COLLECTION = {nom de domaine, libelle domaine collect}
DOMAINE ÉNUMÉRÉ = {valeurs domaine}
DOMAINE PREDEFINI = {nom de domaine, type domaine}

TRANSITION D'ETAT = {Numéro de transition d'état, nom d'opération, nom de classe état1, nom de classe d'état2}
DÉCLENCHEUR = {facteur pour, condition si, nom événement, nom d'opération}
CLASSE D'ETAT CORRÉLÉ = {nom de classe d'état, nom de classe}
CLASSE D'ETAT CALCULE = {nom de classe d'état, nom de classe}
ÉVÉNEMENT = {nom événement, nom de classe}
ÉVÉNEMENT INTERNE = {nom événement, prédicat}
ÉVÉNEMENT EXTERNE = {nom d' événement, message}
ÉVÉNEMENT TEMPOREL = { nom événement, NomPredicatTemporel }
LIEN DE COMP CL = {numéro de lien stat[3], nom de lien, nom de classe1, nom de classe2, valuation}
LIEN DE REF = {numéro de lien stat, nom de lien, nom de classe1, nom de classe2, valuation}
LIEN D'HÉRITAGE = {numéro de lien stat, nom de classe1, nom de classe2}
LIEN STAT D'ATTR = {numéro de lien stat, nom de classe1, nom de propriété}
ASSERTION = {texte d'assertion, nom de classe}
CONTRAINTE D'ATTRIBUT = { texte d'assertion}
GTE_TRANSITION D'ÉTAT = {nom de GTE, nom de transition d'état}
ASSERTION_LIEN STAT D'ATTR = {texte d'assertion, numéro lien stat}
ASSERTION_LIEN STAT_CLASSE = {texte d'assertion, numéro lien stat}

L'utilisation de numéro-de-lien-stat comme identifiant de la table associée à la classe LIEN_STATIQUE oblige à ajouter cet attribut au niveau de toutes les tables dont la classe correspondante est héritière de la classe LIEN_STATIQUE.

c) Application des règles d'optimisation

1)
ÉVÉNEMENT = {nom événement, NomClasse}
ÉVÉNEMENT INTERNE = {nom événement, prédicat}
ÉVÉNEMENT EXTERNE = {nom événement, message}
ÉVÉNEMENT TEMPOREL = {nom événement, NomPredicatTemporel }

donnent la table

ÉVÉNEMENT = {nom événement, NomClasse, message_prédicat, Type}

2)
LIEN DE COMP CL = {numéro de lien stat, nom de lien, nom de classe1, nom de classe2, valuation}
LIEN DE REF = {numéro de lien stat, nom de lien, nom de classe1, nom de classe2, valuation}
LIEN D'HÉRITAGE = {numéro de lien stat, nom de classe1, nom de classe2}

[3] Nous avons ajouté un numéro de lien statique parce qu'il n'y avait pas de clause unique dans toute la hiérarchie d'héritage

donnent la table
LIEN STAT.CLASSE = {numéro de lien stat, nom de lien, nom de classe1, nom de classe2, valuation, Type}

3)
CLASSE D'ÉTAT CORRÉLÉ = {nom de classe d'état, nom de classe}
CLASSE D'ÉTAT CALCULE = {nom de classe d'état, nom de classe}
donnent
CLASSE D'ÉTAT = {nom de classe d'état, nom de classe,Type}

4)
ASSERTION_LIEN STAT D'ATTR = {texte d'assertion, numéro lien stat}
ASSERTION_LIEN STAT_CLASSE = {texte d'assertion, numéro lien stat}
donnent
ASSERTION_LIEN STAT= {texte d'assertion , numéro lien stat ,type lien stat}

5)
ASSERTION = {texte d'assertion, nom de classe}
CONTRAINTE D'ATTRIBUT = { texte d'assertion}
donnent
ASSERTION = {texte d'assertion, nom de classe, Type}

d) Schéma optimisé
CLASSE = {nom de classe}
GTE = {nom de GTE, nom de classe}
PROPRIÉTÉ = {nom de propriété, nom de domaine, nom de classe}
OPÉRATION = {nom d'opération, nom de classe, spécification d'opération}
COMPOS DOMAINE = {nom de domaine, libelle domaine agrégat, libellé composant domaine agrégat}
DOMAINE COLLECTION = {nom de domaine, libelle domaine collect}
DOMAINE ÉNUMÉRÉ = {valeurs domaine}
DOMAINE PREDEFINI = {nom de domaine, type domaine}
TRANSITION D'ÉTAT = {Nom de transition d'état, nom d'opération, nom de classe état1, nom de classe état2}
DÉCLENCHEUR = {facteur pour, condition si, nom d'événement, nom d'opération}
CLASSE D'ÉTAT = {nom de classe d'état, nom de classe,Type}
ÉVÉNEMENT = {nom événement, nom de classe,message_prédicat, Type}
LIEN STAT.CLASSE = {numéro de lien stat, nom de lien, nom de classe1, nom de classe2, valuation,Type}
LIEN STAT D'ATTR = {numéro de lien stat, nom de classe1, nom de propriété}
ASSERTION = {texte d'assertion, nom de classe, Type}
GTE_TRANSITION D'ETAT = {nom de GTE, nom de transition d'état}
ASSERTION_LIEN STAT = {texte d'assertion, numéro lien stat, type lien stat}

NB :
Dans les tables ainsi définies, nous aurons à rajouter les informations graphiques qui permettent la reconstitution de la représentation sur l'écran.
Il s'agira :
- pour la classe
    abscisse et ordonnée du coin supérieur gauche.
    abscisse et ordonnée du coin inférieur droit.
- pour le lien
    dans le graphe statique (les liens sont représentés par des flèches)
    abscisse et ordonnée des points origine et extrémité

    dans le schéma local de classe (les liens sont représentés par des rectangles)
    abscisse et ordonnée du coin supérieur gauche
    abscisse et ordonnée du coin inférieur droit.

4.2 - analyse globale de l'interface C++/Tables Relationnelles

4.2.1- Introduction

Nous pouvons noter deux options d'interfaçage.
La première consistera à opérer entre la structure interne et les tables relationnelles, comme l'indique le schéma suivant (Figure 4.3) :

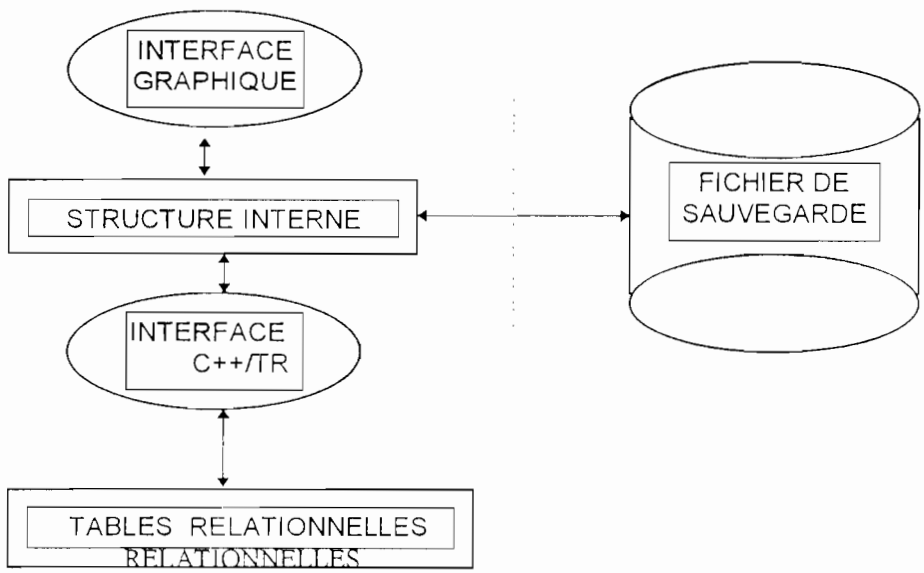

Figure 4.3 La solution transitoire de test de la couche d'interfaçage

En fait, cette première solution n'est qu'une option transitoire. Elle sert à tester la couche d'interfaçage qui fait l'objet de ce chapitre.

Parallèlement au mode de traitement antérieur, le schéma relationnel sera mis en place et géré par une couche interface. A terme, le système de fichier déjà en place sera définitivement remplacé par les tables relationnelles.
La seconde solution est une amélioration de la première. Elle consiste à se passer de la structure interne. Elle peut se schématiser comme suit (Figure 4.4) :

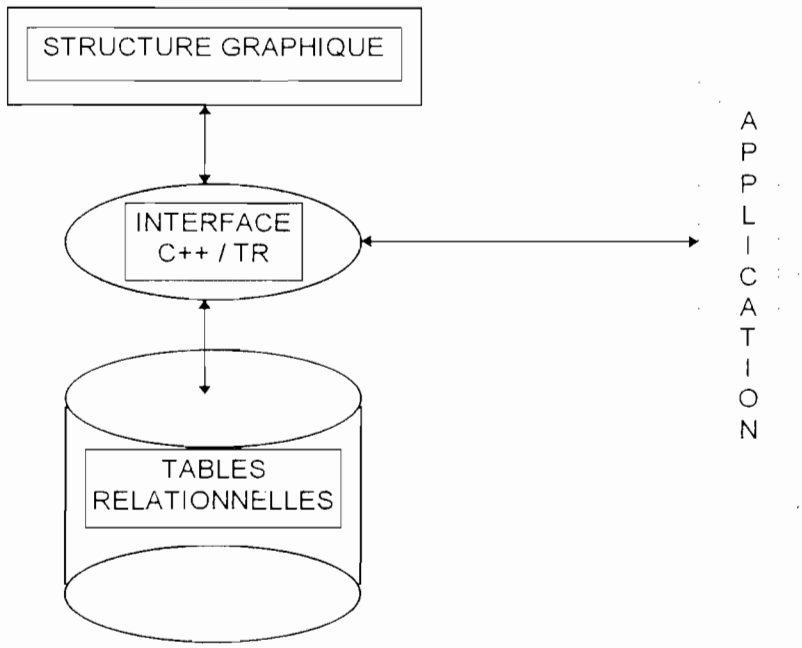

Figure 4.4 : la solution finale

La seconde solution à l'avantage d'éviter le problème du swapping.
Toutefois nous jugeons qu'une bonne implémentation de l'interface devra pouvoir s'adapter à toutes les deux solutions. Ainsi, cette interface sera testée dans un premier temps avec la première solution. Et à l'issue de ces tests, nous allons l'adapter à la deuxième solution pour pallier au problème du swapping.

4.2.2 Gestion de la métabase

4.2.2.1 - Création d'une classe

Dans une classe, tout est lien.
La création d'une classe consistera à instancier des liens au niveau des tables lien du métamodèle. De plus, il faut ajouter une ligne de nom de table dans la métatable CLASSE.

4.2.2.2 - Suppression d' une classe

Dans ce cas, il faut s'assurer que la classe n'a aucun lien. La cas échéant, il faut demander à supprimer tous les liens dans lesquels elle participe. La suppression consiste alors à enlever le nom de la table correspondant à la classe de la métatable CLASSE.

4.2.2.3 - Modification d'une classe

Cette modification portera, selon les cas sur une ou plusieurs des métatables suivantes :

- CLASSE,
- OPÉRATION,
- CLASSE D'ÉTAT,
- ÉVÉNEMENT,
- LIEN STAT.CLASSE,
- LIEN STAT D'ATTR,
- ASSERTION,
- PROPRIÉTÉ

4.2.2.4 - Création d'un lien

Vérifier si les tables associées à la classe origine et la classe extrémité existent, dans ce cas, créer une ligne dans la métatable correspondante au type de lien (LIEN STAT.CLASSE, LIEN STAT.D'ATTR).
La création d'un lien nécessite de connaître les coordonnées de l'origine et de extrémité du lien. Il suffira, alors d'appliquer les règles de transformation entre la classe origine et la classe extrémité selon le type de lien et la valuation.
La fonction créer_lien prendra en paramètre les coordonnées de l'origine et de l'extrémité du lien, la classe origine, la classe extrémité, le nom du lien, la valuation et le type du lien.
La primitive du SGBD Paradox utilisée pour insérer des enregistrements dans une table est :

int PXRECAPPEND.

4.2.2.5 - Suppression d'un lien

Supprimer une instance de lien dans la table du lien correspondant (LIEN STAT.CLASSE ou LIEN STAT. D'ATTR).

Primitive de positionnement : int PXRECFIRST, int PXRECLAST
Primitive de suppression d'un enregistrement : int PXRECDELETE

4.2.2.6 - Modification d'un lien

c'est la mise à jour d'une ligne des métatables (LIEN STAT.CLASSE ou LIEN STAT. D'ATTR).

La primitive à utiliser est int PXRECUPDATE.

4.3 Génération des schémas de l'application finale

A ce niveau, les actions qui seront implémentées consisteront à muter l'ensemble des classes d'une application en tables relationnelles en se basant sur notre logique de transformation.

4.3.1 Spécification des opérations
Cas des liens

Pour caractériser un lien nous avons besoin en particulier du nom de la classe origine, de la classe extrémité, du nom du lien, de son type et de la valuation :
L'objet lien peut donc se définir comme suit :

propriétés
- nom_lien:
- classe origine:
- classe-extrémité:
- valuation:
- type-lien:
- abscisse_origine_lien:
- ordonne_origine_lien:
- abscisse_extrémité_lien:
- origine_extrémité_lien:

méthodes
- créer_lien(constructeur)
- modifier_lien()
- supprimer_lien()

4.3.1.1 Création d'un lien

a) cas d'un lien de composition
ajouter l'identifiant de la table associées à la classe origine dans les attributs de la table associée à la classe extrémité.

Primitive de restructuration de la table annoncée mais non spécifiée.

b) cas d'un lien de référence simple
- ajouter l'identifiant de la table associée à la classe extrémité dans les attributs de la table associée à la classe origine.

Primitive de restructuration de table annoncée mais non spécifiée.

c) cas d'un lien de référence multiple

- créer une table contenant les identifiants des tables associées aux classes origine et extrémité.

Primitive : int PXTBLCREATE

d) cas d'un lien d'héritage:

- ajouter l'identifiant de la table associée à la classe extrémité comme attribut de la table associée à la classe origine

Primitive de restructuration annoncée mais non spécifiée.

- Pour tous les liens de référence simples ayant comme origine la classe extrémité, créer un lien de référence entre la classe origine issue de l'héritage et ces classes.

Pour tous les liens de composition ayant comme extrémité la classe extrémité, créer un lien de composition avec la classe origine issue de l'héritage et ces classes..

N.B. Pour les liens qui sont rajoutés à partir des relations d'héritage ils doivent porter le même numéro de lien que la relation d'héritage; ceci pour garantir la cohérence lors de la suppression d'une relation d'héritage.

e) cas d'un lien statique d'attribut

- ajouter le nom de propriété aux attributs de la table correspondant à la classe.

Primitive de restructuration annoncée mais non spécifiée.

4.3.1.2 Suppression d'un lien

a) cas d'un lien de référence simple

-supprimer l'identifiant de la table associée à la classe extrémité de la table associée à la classe origine ;

Primitive de restructuration annoncée mais non spécifiée

b) cas d'un lien de référence multiple

- supprimer la table contenant les identifiants des tables associées aux classes origine et extrémité.

Primitive : int PXTBLDELETE

c) cas d'un lien d'héritage

- suppression des champs correspondant à la transformation des liens de référence et de composition dans la table associée à la classe spécialisée.

-supprimer l'identifiant de l'extrémité comme attribut de la table origine

Primitives de restructuration annoncée mais non spécifiée

d) cas d'un lien statique d'attribut

- supprimer le nom de la propriété dans la table correspondant à la classe.

Primitive de restructuration annoncée mais non spécifiée

4.3.1.3 modification d'un lien

- modification portant sur le type de lien.

Si la modification porte sur le type de lien et/ou sur la valuation alors :

- supprimer le lien (utiliser l'opération supprimer_lien)
- créer le nouveau lien (utiliser l'opération créer_lien)

- modification portant sur les classes origines ou extrémité d'un lien (Figure 4.5)

TYPE	Simple	multiple
HÉRITAGE	supprimer le lien et le recréer	
COMPOSITION	Sur l'origine : supprimer l'identifiant dans l'extrémité. Ajouter l'identifiant de la nouvelle origine dans l'extrémité Sur l'extrémité : supprimer l'identifiant de l'origine dans l'ancienne extrémité et le reporter sur la nouvelle extrémité.	
RÉFÉRENCE	Sur l'origine : Supprimer l'identifiant de l'extrémité dans l'ancienne origine et le reporter dans la nouvelle origine. Sur l'extrémité : Supprimer l'identifiant de l'ancienne extrémité dans la table origine et reporter l'identifiant de la nouvelle extrémité.	Modifier l'identifiant de l'origine ou de l'extrémité dans la troisième table créée pour le lien.

Figure 4.5 Modification portant sur les classes origine ou extrémité d'un lien

Lien statique d'attribut :

- modification portant sur l'extrémité:

modifier la propriété en cas de changement du nom, effectuer la correction dans la classe origine.

Primitive : int PXRECUPDATE

changer l'identifiant de la propriété dans la classe origine

Primitive de restructuration de table annoncée mais non spécifiée.

- modification portant sur l'origine

Supprimer la propriété dans l'ancienne classe origine et la rajouter dans la nouvelle classe origine.

Primitive de restructuration de table annoncée mais non spécifiée.

Dans tous les cas mettre à jour la table de lien correspondant.

Primitive : int PXRECUPDATE

N.B.

Toute modification doit induire une mise à jour de la métabase

4.3.2 Spécification des opérations

Cas des classes

4.3.2.1 Création d'une classe

Il suffit de créer une table portant le nom de la classe et ajouter le nom de la table dans la metatable CLASSE

Ensuite, faire appel à l'opération créer_lien pour tout lien concernant cette classe.

Les métatables à utiliser sont :

- CLASSE,
- LIEN STAT CLASSE,
- LIEN STAT D'ATTR.

Les primitives Paradox requises sont :

- int PXTBLCREATE
- la primitive de restructuration de table.est annoncée mais non spécifiée

4.3.2.2 Suppression d'une classe

Nous pouvons remarquer qu'une classe est définie par un ensemble de liens. Donc supprimer une classe correspond à un ensemble de suppressions de liens; précisément, ceux définissant la classe.

Il ne faut pas oublier de supprimer le nom de la classe dans la métatable CLASSE.

4.3.2.3 Modification d'une classe

Si la modification porte sur le nom alors changer le nom de la table correspondant à l'aide de la primitive int PXTBLRENAME.

Si la modification porte sur les liens se référer aux procédures de modification de liens.

4.3.3 Procédures de pré-insertion et de pré-suppression

Elles se rapportent particulièrement aux cas d'héritage. En fait, les règles de transformation en table ne traduisent pas toujours les contraintes d'héritage.

Ces procédures seront donc associées aux tables correspondantes aux classes spécialisées ou généralisées pour gérer de telles contraintes.

4.3.3.1 Cas d'une couverture

C'est surtout la suppression qui pose problème à ce niveau car la suppression d'un objet spécialisé devra entraîner celle d'un objet généralisé.

4.3.3.2 cas de la disjonction

- A ce niveau, c'est précisément l'insertion qui pose problème.

Avant d'insérer une clé dans une table spécialisée, il faut s'assurer que :

- cette clé ne figure dans aucune des autres tables spécialisées
- cette clé est dans la table généralisée.

4.3.3.3 cas de la partition

Il faut remarquer qu'une partition est à la fois une disjonction et une couverture. De ce fait, pour la gérer il faut utiliser la procédure de pré-insertion qui gère la disjonction et la procédure de pré-suppression qui gère la couverture.

4.3.4 Interface entre les tables relationnelles d'une application et les classes C++

Les opérations communes à toutes les classes sont implémentées au niveau d'une classe interface généralisée. L'objectif étant de rendre transparent l'utilisation des tables relationnelles, il est donc nécessaire d'implémenter des méthodes qui permettent d'instancier une classe C++ à partir des tables, et de sauvegarder une classe existante. Ainsi la dynamique de l'interface globale sera modélisée comme suit (Figure 4.6) :

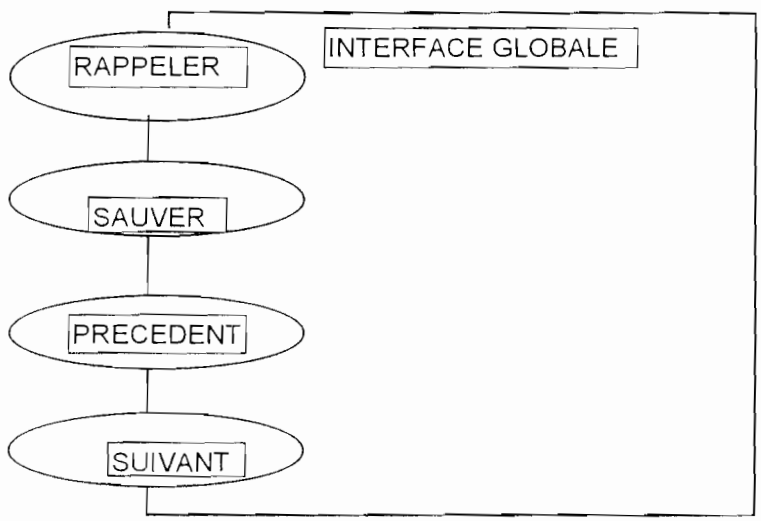

Figure 4.6 : La classe interface généralisée

- Rappeler : permet d'instancier un objet C++ à partir des tables relationnelles.
- Sauver : sauve sur disque une instance d'objet C++.
-Précédent : permet de revenir à la classe précédente.
- Suivant : permet de passer à la classe suivante.

En plus de ces méthodes généralement utilisées par les objets, la dynamique des objets se modélise de la manière suivante (Figure 4.7) :

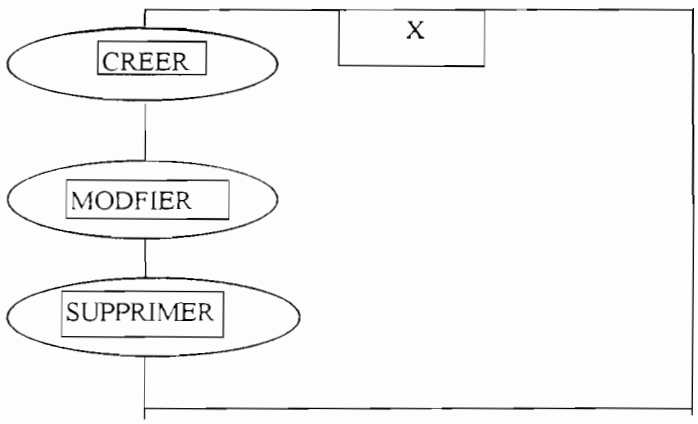

Figure 4.7 : la dynamique des objets

La spécification de ces méthodes a été faite dans la partie Génération du schéma de l'application finale.

*Chapitre 5 : L'interface d'aide à la saisie de concepts O**

5.1 Fonctionnalités

Le prototypeur doit partir d'un schéma conceptuel valide par rapport à la méthode. Ce qui nous a conduit à réaliser d'abord une interface d'aide à la spécification de schémas conceptuels basés sur le modèle de la méthode O*. Elle fournit des éditeurs graphiques adaptés à l'utilisation de formalismes graphiques du modèle (graphe statique, schéma local de classe, graphe de transition d'état, graphe dynamique). Elle assure la cohérence entre les différents graphes qui sont produits et permet d'automatiser les contrôles de validation du schéma conceptuel.
O* propose deux formalismes de description du schéma conceptuel, l'un est graphique, l'autre est textuel. Les deux expressions graphique et textuelle sont équivalentes et complémentaires et notre interface permet de l'assurer. En effet la métabase qui stocke les éléments du schéma conceptuel peut être construite de façon équivalente à l'aide du formalisme graphique ou textuel : un programme de traduction obtenu à partir de générateurs d'analyse lexicale et syntaxique permet d'assurer la traduction d'un texte O* en une métastructure et réciproquement [Arial D / 95-1]
A ce stade de notre recherche, nous avons réalisé le schéma local de classe et le graphe statique comme le montreront les figures qui suivent. Le schéma de classe, qui contient toutes les caractéristiques locales à une classe (Figure 5.1). Ce type de schéma concerne l'étape 3 du processus de modélisation. Le nom de classe étudié (aaaa dans l'exemple) est entouré en gras. Au-dessous sont placées les différentes caractéristiques pouvant être définies sur la classe :

- les classes généralisées desquelles la classe étudiée hérite (classe bbbb)
-les propriétés à valeur domaine NomProp1 à valeur NomDom1 et NomProp2 à valeur NomDom2),
- les propriétés à valeur dans une classe (les propriétés à valeur multiple sont symbolisées par une triple bordure)
-les références (les propriétés à valeur multiple sont symbolisées par une triple bordure)
-les opérations
-les événements (invisibles sur le schéma à cause de la taille limitée de l'écran)
-les contraintes statiques (d'attribut,d'unicité ou d'héritage) (invisibles sur le schéma à cause de la taille limitée de l'écran)
-les commentaires (invisibles sur le schéma à cause de la taille limitée de l'écran)
Les opérations applicables à la classe sont accessibles dans la barre de menu. On accède aux opérations applicables à chaque élément du schéma par des menus surgissants (pop up) qui sont affichés dès que l'on clique sur un élément.

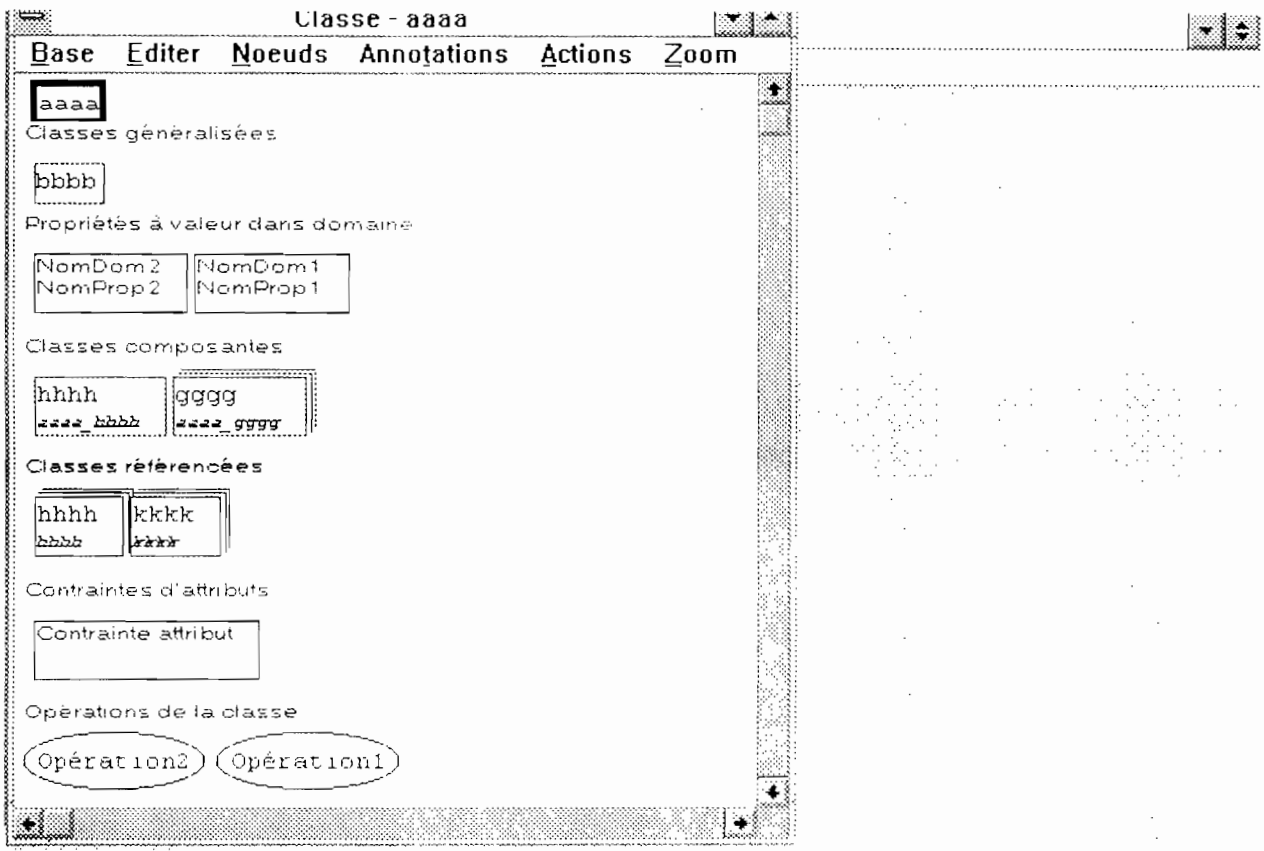

Figure 5.1: le schéma de classe

Le graphe statique (Figure 5.2) qui permet d'éditer les liens de composition, de référence et d'héritage reliant les classes. Ce type de graphe est un support à l'étape 4 du processus de modélisation. Un schéma conceptuel comporte un seul graphe statique. On distingue les différents liens par leur couleur : lien d'héritage en vert, lien de référence en bleu et lien de composition en rouge. Une barre d'outils permet de sélectionner le type de lien que l'on veut insérer. Un clic sur un des éléments (classe ou lien) du graphe statique fait apparaître un menu surgissant (pop up) contenant des opérations agissant sur l'élément lui même ou faisant transfèrer vers un autre type de fenêtre (schéma de la classe par exemple). Par exemple un clic sur un lien de composition ou de référence fait apparaître le menu pop up contenant les items Supprimer et Renommer.

Une ligne d'état au bas de la figure permet de guider le développeur en lui spécifiant l'action en cours ou la prochaine action à faire vu le contexte actuel.

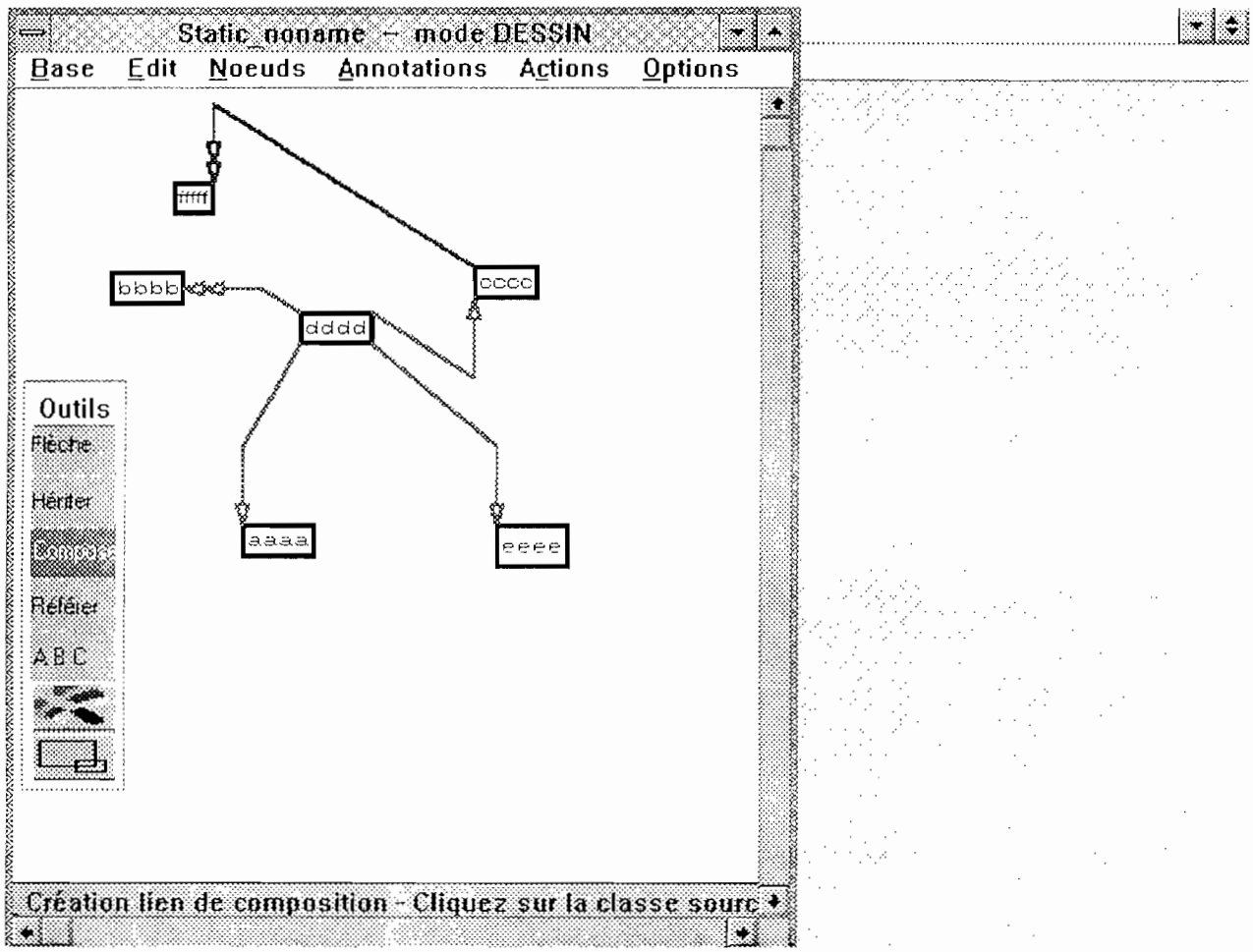

Figure 5.2 le graphe statique

Des contrôles de validation sont effectuées automatiquement par l'outil : par exemple l'unicité du nom de classe et l'interdiction des cycles dans les graphe de composition, d'héritage et dans le graphe dont les arcs sont des liens d'héritage et des liens de composition est assuré (figure 5.3 et figure 5.4) .Pour faciliter les manipulations de redimensionnement nous avons rendu mobiles les classes et les liens graphiques.

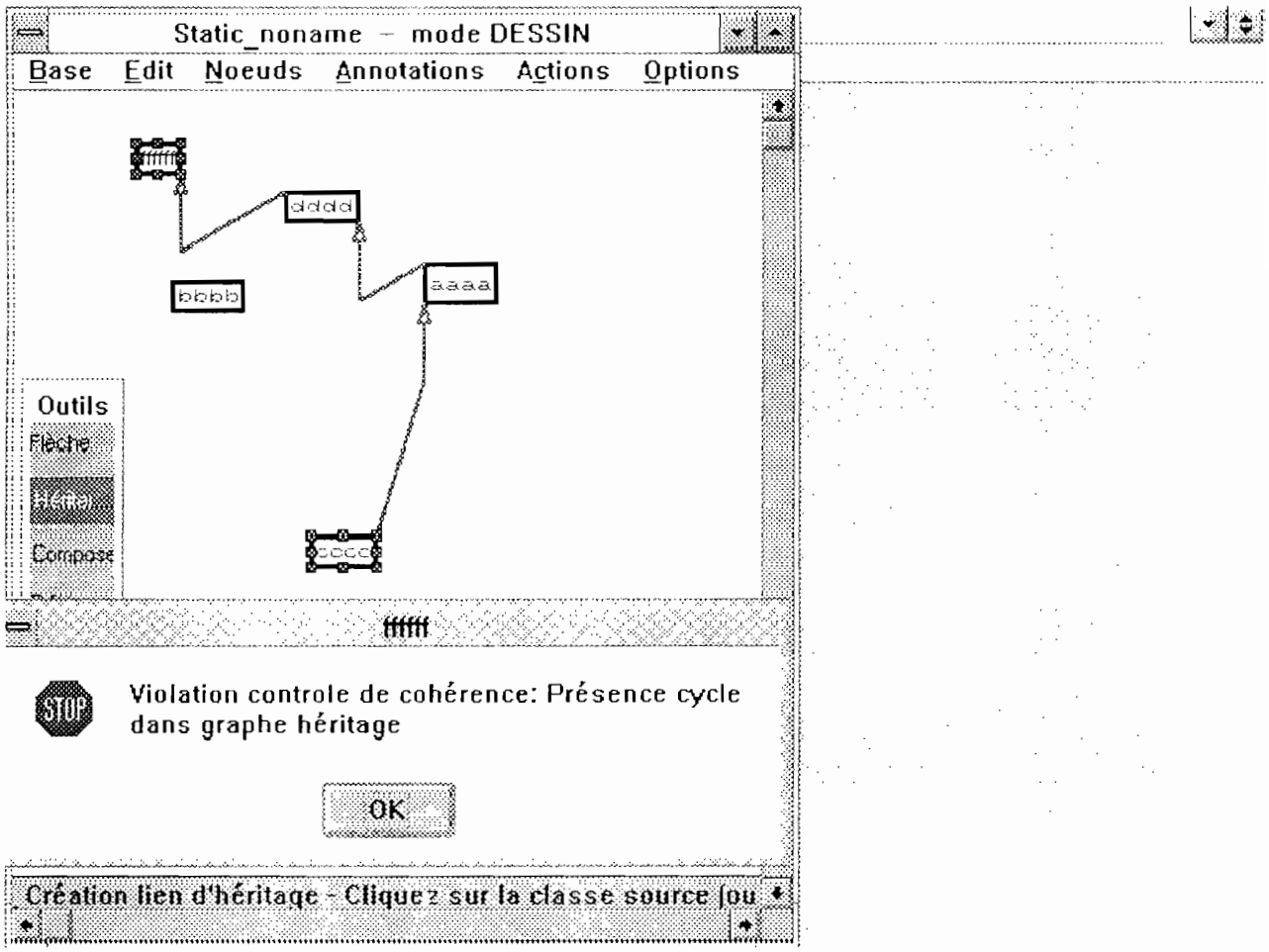

Figure 5.3 : Violation contrôle de cohérence (Cycle dans le graphe d'héritage)

Dans la figure 5.3 le développeur tentait de créer un lien d'héritage ffffff hérite de cccc. Dans la figure 5.4 l'outil lui indique le cycle détecté.

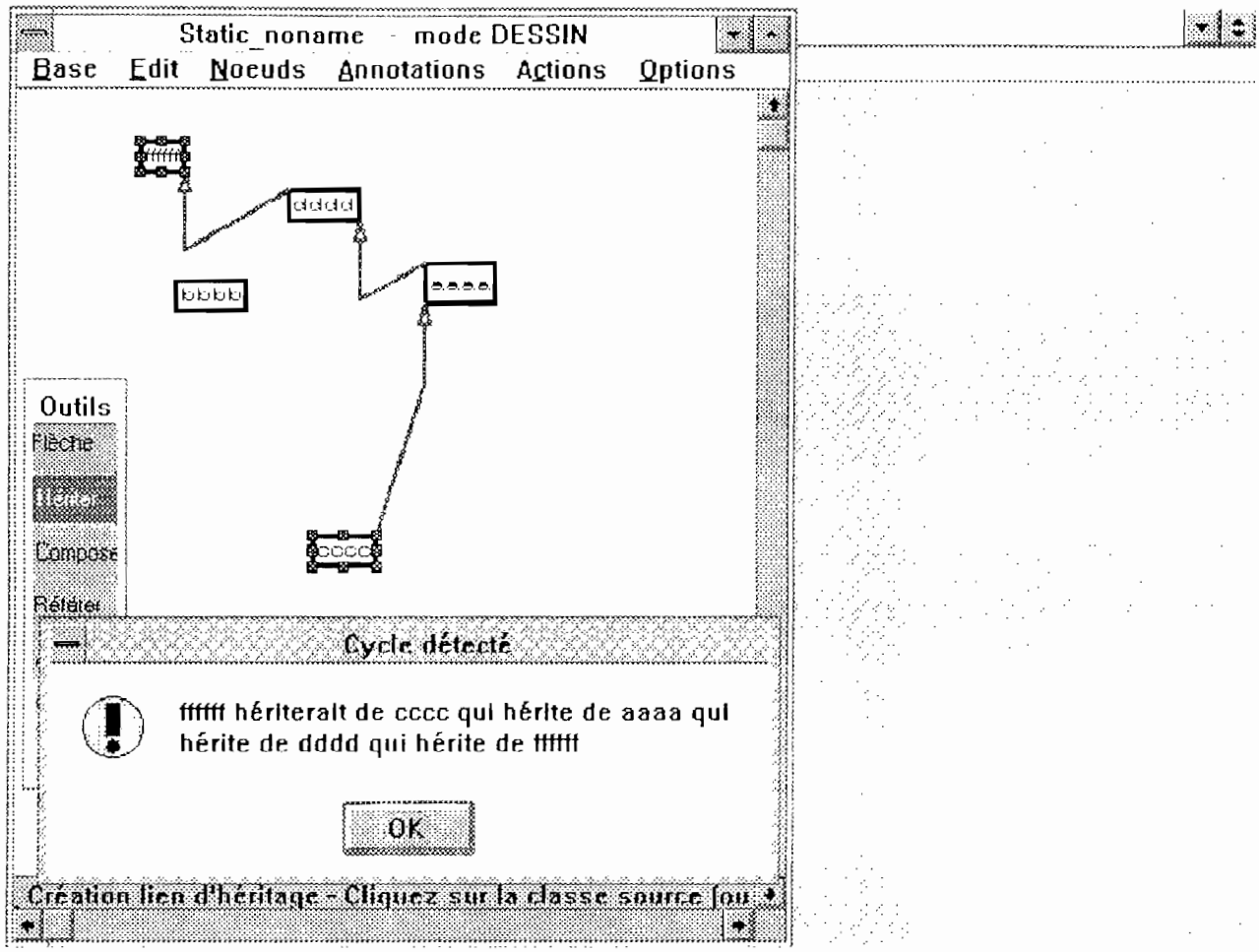

Figure 5.4 : le cycle détecté

Enfin, un dernier type de fenêtre (figure 5.5) permet d'accèder aux schémas de classe, au graphe statique et aux graphes dynamiques du schéma conceptuel. Il sert de menu général, par lequel les opérations concernant l'ensemble du schéma conceptuel peuvent être effectués : les contrôles de validation, la génération de la spécification textuelle, etc. On peut constater sur la figure que les différentes fenêtres peuvent être iconifiées, ce qui perrmet de passer rapidement d'un graphe à un autre.

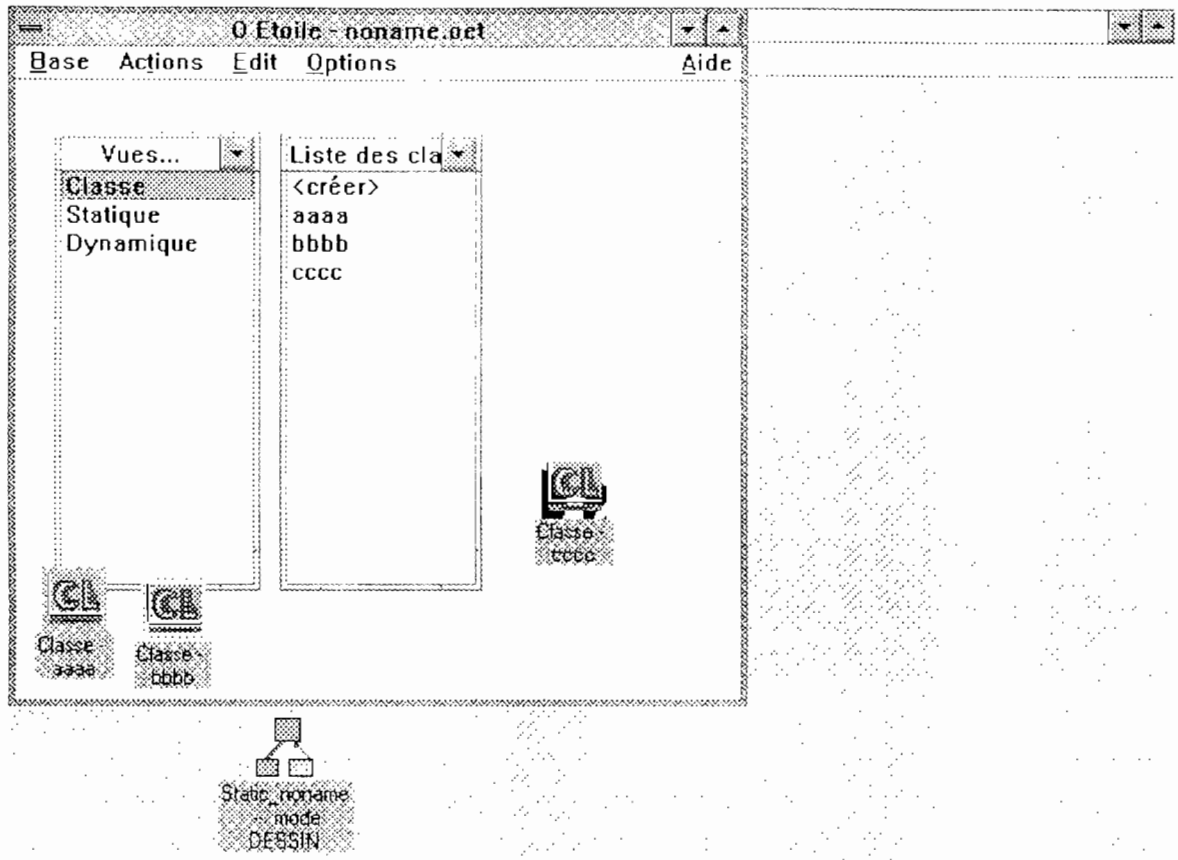

Figure 5.5 : le menu général

5.2 Réalisation

5.2.1 Structures de données

Pour les classes nous avons défini une classe TDicoClasses qui hérite de la classe Dictionnary de la librairie de classes CLASSLIB de Borland C++ qui permet de gérer

-- la création d'une nouvelle d'une classe,
-- la supppression et la modification d'une classe,
-- le renommage d'une classe,
-- dire si une classe appartient au dictionnaire ou pas,
-- de détacher une classe du dictionnaire sans la tuer,
-- sauver ou de lire l'ensemble des classes à partir du disque
-- etc

La structure de données Dictionnary est un dictionnaire qui est un ensemble d'associations ; chaque association étant un couple clé, valeur. Ici les clés sont les noms des classes et les valeurs les objets TSchemaClasse eux même (dans TSchemaClasse T signifie type)

Au niveau des éléments du modèle O*, nous distinguons le conceptuel du graphique si l'élément doit être visualisable à l' écran. Ainsi à la classe TSchemaClasse nous assssocions la classe TFenSchemaClassse, le lien étant réciproque ;
La TSchémaClasse connait la TFenSchemaClassse à laquelle elle est asssociée par l'intermédiaire d'un pointeur FenSchemaClassse membre de TSchémaClassse et la TFenSchémaClassse connait la TSchémaClassse à la quelle elle est asssocieé par l'intermédiaire d'un pointeur membre de TSchemaClassse pSchemaClasse (Figure 5.6).

Ceci reste valable pour la classe TGrapheStatique à laquelle est associée la classe TFenGrapheStatique. Les classes TFenGrapheStatique et TFenSchemaClasse servent à intercepter les événements Windows venant de l'utilisateur et de l'environnement pour les répercuter au niveau des classes conceptuelles ou d'effectuer une action graphique donnée. Les classes TFenSchemaClasse et TFenGrapheStatique sont dérivées de la classe TWindow qui fait partie de la librairie de classe OWL 2 de Borland C++

```
class TSchemaClasse : public Sortable , public TStreamable
{
//*********************************************************************
                        ++++++++++++++++++PUBLIQUE+++++++++++++++
public :
        char NomClasse[LGMAXNOM] ;
        TListeProprietesDsDomaine * pListeDePropDsDomaine ;
        TSetAssertions          * pEnsembleAssert_Attribut;
        TEnsembleConcept        * pEnsembleAssert_Unicite;
        TEnsembleConcept        * pEnsembleAssert_Heritage;
        TCommentaire            * pCommentaire ;
        TEnsembleConcept* pEnsembleOperations ;
        TEnsembleConcept* pEnsembleEvenements ;

        TFenSchemaClasse * FenSchemaClasse ;


//*****************************************************************************************************


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
        TSchemaClasse ( LPSTR) ; // constructeur

///[]->Methodes d'information
        //virtual LPSTR GetClassName () ;
//.....................................................................
        void RenommerClasse(LPSTR);
//.....................................................................
} ;
```

```
class TFenSchemaClasse : public  TModeleGraphique
{

//@ @ @ @ @ @ @ @ @ @  Liste des donnees
        TSchemaClasse* pSchemaClasse ; // SchémaClasse associée à
                                       //cette objet Fenetre

        //**********************
        TFenSchemaClasse(PTWindowsObject,LPSTR,TSchemaClasse* ) ;

//.....
//{{{{{{{{{{{ Methodes de reponses a un evenement
void Trt_SaisieCommentaire(RTMessage) = [CM_FIRST+IDM_ADD_COMMENT];
void Trt_AjouterLienHeritage(RTMessage) = [CM_FIRST+IDM_ADD_HERIT];
voidTrt_AjouterLienComposition(RTMessage)=[CM_FIRST+IDM_ADD_COMP];
void Trt_AjouterLienReference(RTMessage) =[CM_FIRST+IDM_ADD_REFER];

void Trt_AjouterAssertAttribut(RTMessage) =[CM_FIRST+IDM_ADD_ASATTR];
void Trt_AjouterAssertUnicite(RTMessage) =[CM_FIRST+IDM_ADD_ASUNIC];
void Trt_AjouterAssertHeritage(RTMessage) =[CM_FIRST+IDM_ADD_ASHERIT];
void Trt_AjouterEvenement(RTMessage) =[CM_FIRST+IDM_ADD_EVT];
//

//~~~~ menu flottant suite à un clic sur une classe composante

void RenommerLibelleLienCompos(RTMessage)
                                        = [CM_FIRST + IDM_LIENCOMPOREN] ;
void SupprimerLienCompos(RTMessage)
                                        = [CM_FIRST + IDM_LIENCOMPODROP] ;
void EditerClasseExtrLienCompos(RTMessage)
                                        = [CM_FIRST + IDM_LIENCOMPOEDIT] ;
//.....

} ;
```

Figure 5.6 Une partie de la TSchemaClasse et de la classe TFenSchemaClasse en C++

Pour les liens, nous avons défini une structure de données appelée TConteneurDeLiens qui est composée d'un dictionnaire d'entrée et d'un tableau de liens (Figure 5.7 et 5.8). Les clés dans le dictionnaire d'entrée sont les noms de classe et la valeur un couple de listes d'indices du tableau de liens. La première liste d'indices correspond à des liens dont la classe (la clé) est origine. La seconde liste d'indices correspond à des liens dont la classe (la clé) est extrémité. Il existe un conteneur de liens pour chaque lien statique (héritage,référence,composition).

Un lien est unique dans le tableau de liens

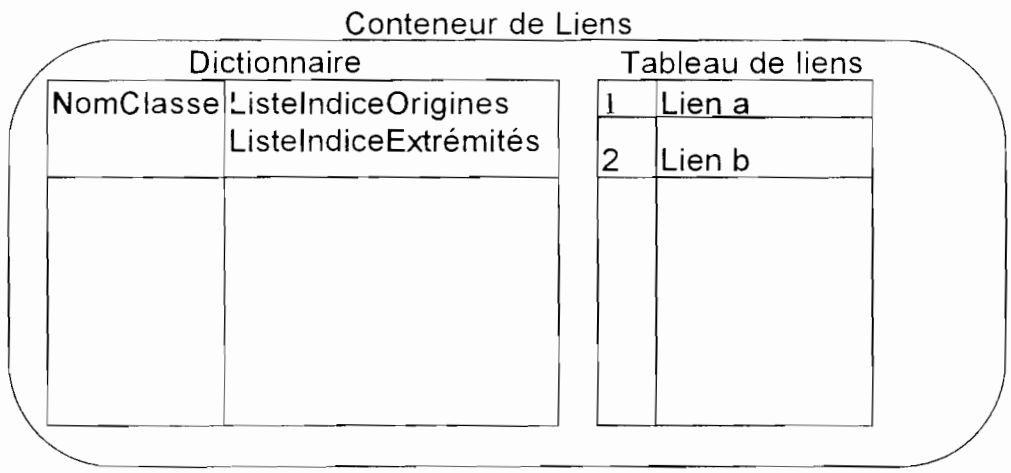

Figure 5.7 Leconteneur de liens

```
class  TConteneurLiens : public TStreamable
{
//                          ------------------PRIVE----------------
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes



//                          +-+-+-+-+-+-+-+-+-+-+PROTEGE+-+-+-+-+-+-+-+-+
protected:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees
TDicoEntree  * pdicoEntree ;
TArrayWithHolesOfLinks* ptableauliens ;


//                          ++++++++++++++++++++PUBLIQUE+++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees
//on maintient une liste de liens d'héritage créés  à partir d'une
//TFenSchemaClasse pour pouvoir leur donner des points origine et
//extrémités de leur lien graphique associé dans TFenGrapheStatique
TListeDeLiens*  pLiensHeritDeTFenSchemaClasse ;
```

```
//on demande au conteneur de liens de maintenir une liste de classe
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage
TListeDeClasses* pClassesJustBeenRenamed ;
//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TConteneurLiens() ;

//[]->Méthodes d'information

TListeDeLiens * LiensDtClasseEstOrigine(LPSTR szlienNomClasse,HWND );
TListeDeLiens * LiensDtClasseEstExtremite(LPSTR NomDeClasse ,HWND);
TListeDeLiens * LiensDeLaClasse(LPSTR NomDeClasse, HWND);
//BOOL   LienEstMembre(TSchemaClasse * , TSchemaClasse *,HWND) ;
TLienConceptuel   *        IdentifierLien(RTMessage) ;

//->[]Methodes de manipulation
void SupprimerLienConceptuel(TLienConceptuel *,HWND) ;
//void DetruireClasseEtDependances(LPSTR,HWND) ;

void AjouterLienHeritage(LPSTR,LPSTR,POINT,POINT, TPosLienGrSurIcone,
                                    TPosLienGrSurIcone,HWND)
                    ;//méthode surchargée II
void AjouterLienConceptuel(TLienConceptuel * ,POINT,POINT,
                     TPosLienGrSurIcone,
                                    TPosLienGrSurIcone,HWND);

void DisplayAllLinks(HDC) ;

void AdjustGrLinksOnClassesRenamed(HDC) ;

//on demande au conteneur de liens de maintenir une liste de classes
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage)
void AddClassJustRenamedInList(TSchemaClasse * ) ;
int MettreLienDsTableauLiens(TLienConceptuel *,HWND) ;
void MajListesOriginesEtExtremites(LPSTR,LPSTR,int,HWND) ;
//OK<----Methodes d'autorisation
//<< >> << >> Methode de gestion des flux
virtual void write(Ropstream ) ;
virtual Pvoid read(Ripstream ) ;
};
```

Figure 5.8 : Une partie de la classeTConteneurLiens en C++

5.2.2 Création d'une instance de classe

Le constructeur a comme arguments le nom de la classe à créer et la fenêtre mère de la fenêtre qui sera associée à l'instance de classe. On vérifie qu'il n'existe pas déjà dans le dictionnaire de données (DicoClasse) une association ayant pour clé le nom de la classe à créer.Si oui la création est avortée après un message au développeur que la classe existe déjà.
Si non on crée dans la mémoire dynamique une instance de la classe en spécifiant le nom de la classe et l'objet fênêtre mère de l'instance. On prévoit dans le destructeur de la classe une opération de libération de la mémoire au cas l'objet classe venait à être détruit.
Une association ayant pour clé le nom de la classe et pour valeur l'objet classe ainsi défini est créé et elle est inséré dans le dictionnaire de classe grâce à une méthode héritée de la classe Dictionnary de la librairie OWL2 de Borland C++.
On crée en même temps une association dans chaque dictionnaire d'entrée des Conteneurs de le liens (héritage, référence, composition) ayant comme clé le nom de la classe et comme valeur une paire de liste d'indices vides ; ceci en prévision d'une création de lien future avec la classe nouvellement crée.

5.2.3 Suppression d'une instance de classe

On fait appel à une fonction prédéfinie de la classe Dictionnary (lookup) qui à partir du nom de la classe retrouve l'association correspondante. On enlève l'association du dictionnaire de classe grâce à une méthode (detach) prévue à cet effet.
Nous détruisons aussi tous les liens statiques associées à cette classe grâce à des méthodes DetruireClasseEtDependances de la classe TConteneurLiens.

5.2.4 Modification d'une instance de classe

Selon le paradigme objet, les données membres sont modifiées (et lues d'ailleurs) par l'intermédiaire de méthodes prévues à cet effet dans la classe de l'objet.
Le renommage d'une classe entraine le changement de nom dans le dictionnaire des classes mais aussi dans le dictionnaire d'entrée de chaque conteneur de liens (héritage,composition,référence).

5.2.5 Création d'une instance de lien

Le constructeur de lien a comme argument un pointeur vers la classe origine et un pointeur vers la classe extrémité. Pour le lien de composition et de référence. il faut renseigner aussi le nom de la propriété et la valuation. Pour tous les trois types de lien, il faut aussi instancier le TLienGraphique associée.
On vérifie que le lien n'existe pas déjà dans le tableau des liens. On vérifie aussi que la classe extrémité figure dans le dictionnaire de classes. Sinon on demande au développeur s'il veut créer une nouvelle classe. Si oui la classe et le lien sont créés; sinon un message « lien non crée » est affiché.

N.B.1 Pour créer un lien statique par l'interface graphique de l'outil on part toujours d'une classe origine existante soit dans le graphe statique par un clic dans le rectangle d'une classe
soit par l'intermédiaire de la fenêtre FenSChemaClasse du schéma de classe local d'une classe préexistante

N.B.2
La classe TLienGraphique contient en particulier les points origine et points extremité des liens graphiques de composition, d'héritage et de référence dans la fenêtre FenGrapheStatique instance de la classe TFenGrapheStatique et les rectangles entourant les icônes des liens graphiques de composition , d'héritage et de référence dans la fenêtre FenSchemaClasse instance de la classe TFenSchemaClasse.

5.2.6 Le conteneur de liens

Le constructeur du conteneur de lien crée en particulier le dictionnaire d'entrée vide et le tableau des liens vide. Il crée en même temps les autres données membres qui sont des intermédiaires dans la gestion graphique des classes et des liens de la fenêtre graphe statique.
Pour insèrer le lien dans le tableau des liens on le fait grâce à une méthode MettreLienDsTableauLiens de TConteneurLIens qui vérifie si le lien n'existe pas déjà dans le tableau lien et appelle la même méthode de TArrayWithHolesOfLinks , qui elle aussi appelle la méthode Ajouter de la même classe. TArrayWithHolesOfLinks est une sous classe de la classe Array de la librairie de classes de Borland C++. La fonction TArrayWithHolesOfLinks ::Ajouter recherche à partir de la borne inférieure un trou où insérer le lien . S'il n'y pas de trous et qu'on ait atteint la borne supérieure une réallocation mémoire est faite. La fonction TArrayWithHolesOfLinks::Ajouter retourne dans tous les cas l'indice du tableau où on a inséré le lien. Dans le dictionnaire d'entrée l'association ayant comme clé le nom de la classe origine du lien voit sa ListeIndicesOrigine enrichie d'un indice, celui retourné par Ajouter. De même l'association du dictionnaire d'entrée ayant comme clé le nom de cla classe extrémité voit sa ListeIndicesExtrémité augmenter du même indice.

5.2.7 Suppression d'un lien

La fonction detach hérité de Array(de la librairie de classes OWL2 de Borland C++) permet de supprimer un lien s'il est présent dans le tableau de liens connaisant son index dans le tableau de liens. La fonction TArrayWithHolesOfLinks:: TrouverIndiceDsTableau(const &tof) retourne l'indice dans le tableau correspondant à l'objet tof ;
Il faut mettre à jour la listeIndicesOrigine de la classe origine qui perd un élément et la listeIndicesExtrémité de la classe extrémité qui perd un élément.

5.2.8 Modification d'un lien

Selon le paradigme objet , les données membres sont modifiées (et lues d'ailleurs) par l'intermédiaire de méthodes prévues à cet effet dans la classe de l'objet.

5.2.9 la persistance des méta-données statiques

a) la sauvegarde

Les données statiques à sauvegarder sont
-l'ensemble des classes contenues dans DicoClasses instance de TDicoClasses
-l'ensemble des liens d'héritage contenues dans ConteneurLiensHERIT instance de TConteneurLiensHERIT
-l'ensemble des liens de composition contenues dans ConteneurLiensCOMPOS instance de TConteneurLiensCOMPOS
--l'ensemble des liens de référence contenues dans ConteneurLiensREFER instance de TConteneurLiensREFER
-l'ensemble des propriétés à valeur dans un domaine contenues dans ConteneurProprietes instance de TConteneurProprietes
-l'ensemble des domaines contenues dans ConteneurDomaines instance de TConteneurDomaines

Les trois classes TConteneurLiensHERIT, TConteneurLiensCOMPOS et TConteneurLiensREFER sont des sous classes de TConteneurLiens..

Pour qu'une classe soit persistante, la librairie Borland C++ sous Windows exige qu'elle dérive de la classe prédéfinie TStreamable et qu'elle soit munie de 5 fonctions membres qui sont :
-un constructeur de la classe ayant comme argument StreamableInit
-une fonction membre de prototype virtual const Pchar streamableName() const ;
-une fonction membre de prototype static PTStreamable build() ;
-une fonction membre de prototype virtual void write(Ropstream) ;
--une fonction membre de prototype virtual Pvoid read(Ripstream) ;
C'est surtout les deux dernières qui sont intéressantes pour le programmeur.
La sauvegarde de toutes les données précédentes se fait par la fonction TFenPrinc::write. TFenPrinc est une classe fenêtre représentant le menu principal et elle dérive de la classe TWindow de la librairie OWL2 de Borland C++ dont les classes sont toutes persistantes.

La déclaration ofpstream os (NomFichier); ouvre un flux os associé à un fichier de nom « NomFichier » en sortie. Il suffit maintenant d'écrire

```
void TFenPrinc::write(Ropstream os)
{
DicoClasses ->write(os) ;
ConteneurLiensHERIT->write(os) ;
ConteneurLiensCOMPOS->write(os);
ConteneurLiensREFER->write(os) ;
ConteneurProprietes->write(os);
ConteneurDomaines->write(os);
}
```

Les classes qui sont des ensembles procèdent par itération sur chacun de leurs éléments (dont la classe est rendue nécessairement persistante par la méthode

précédente) et demandent à chaque élément de se sauvegarder par la syntaxe elément -> write (os), en supposant que elément appartient à l'ensemble.
Pour ces classes ensembles on sauvegarde d'abord le nombre d'éléments, en prévision de la lecture.

Les classes dont certaines données membres doivent persister et qui sont instances de classe (donc nécessairement rendues persistantes par la méthode précédente) doivent demander récursivement à la donnée membre de se sauvegarder par la syntaxe donneeMembre->write(os)
Le processus continue ainsi jusqu'à aboutir à des types prédéfinies pour lesquels la sauvegarde est définie par C++ par la syntaxe os<<donneePredenie.
Pour les éléments de longueur variable comme les chaînes de caractères on sauvegarde d'abord la longueur de la chaîne (ceci en prévision de la lecture) et les caractères de la chaîne sont ensuite sauvegardés un à un par la suite.
Ainsi on abouti à un fichier de nom « NomFichier » contenant les données voulues.

b) la lecture
La déclaration ifpstream is (NomFichier) ; ouvre un flux is associé à un fichier de nom « NomFichier » en entrée.
Ici on recrée en mémoire dynamique les DicoClasses, ConteneurLiensHERIT, ConteneurLiensCOMPOS, ConteneurLiensREFER, ConteneurDomaines, ConteneurProprietes **vides** et on leur demande de se récréer en lisant (fonction membre read) dans le fichier « NomFichier ». Chaque conteneur lit le nombre d'éléments qu'il contient (c'est la première information qui a été sauvegardée) et demande à ses composants de se recréer (fonction membre read). Chaque classe qui possède des données membres persistantes, instances de classes leur demandent de se recréer (fonction membre read) et ainsi récursivemnent jusqu'à aboutissement aux éléments terminaux correspondant aux types prédéfinis qui se recréent en lisant directement dans le fichier par la syntaxe is >> elementTerminal ;
Pour les éléments de longueur variable comme les chaînes de caractères on lit d'abord le nombre de caractères de la chaîne, puis on itère sur ce nombre pour lire chaque caractère du tableau de caractères ainsi recréé.

```
Pvoid TFenPrinc::read(Ripstream is)
{

 DicoClasses    = new TDicoClasses() ;
 ConteneurLiensHERIT= new TConteneurLiensHERIT() ;
 ConteneurLiensCOMPOS = new TConteneurLiensCOMPOS() ;
 ConteneurLiensREFER = new TConteneurLiensREFER () ;
 ConteneurProprietes= new TTableauProprietesDsDom(50,0,10);
 ConteneurDomaines = new TTableauDomaines(50,0,10);

 GrapheStatique = new TGrapheStatique() ;
 Verification = new TVerification() ;

 DicoClasses->read(is);
 ConteneurLiensHERIT->read(is);
```

```
 ConteneurLiensCOMPOS->read(is);
 ConteneurLiensREFER->read(is) ;

 ConteneurProprietes->read(is);
 ConteneurDomaines->read(is);

return this ;
}
```

L'ordre dans lequel les conteneurs sont sauvegardés est important ; ce sera celui de la lecture.

Chapitre 6 : le prototypeur

6.1. Conception et implémentation

Le schéma de principe de l'outil de prototypage est donné à la Figure 6.1

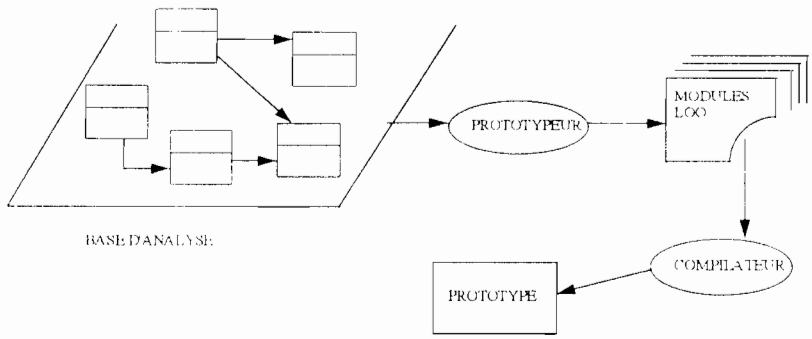

Figure 6.1 : Schéma de principe du prototypeur

Notre outil se situe dès l'analyse dans la perspective d'un processus incrémental de prototypage conduisant au produit final. il comporte donc la production de prototypes. Ces prototypes constituent des moyens privilégiés de communication développeur-client. Ils ont pour objectifs:

- de faciliter l'obtention d'informations sur la pertinence et l'adéquation de la spécification et de la conception du système informatique à venir.
- de servir de précurseur à l'écriture effective du système final [Krief 91].

Du point de vue des fonctionnalités, chaque prototype obtenu devra permettre de :

-créer, modifier et supprimer des instances de classes,

-accéder aux attributs et aux instances de liens de composition ou de référence, et les manipuler,

-saisir les événements externes et simuler les événements temporels du SI,

-suivre la propagation des événements internes dans un graphe dynamique.

Un modèle de conception de dialogue permet à l'outil de générer l'interface utilisateur (Figure 6.2). Pour la définition des écrans, on utilise les éléments de données (attributs et liens statiques) définis dans la métabase.

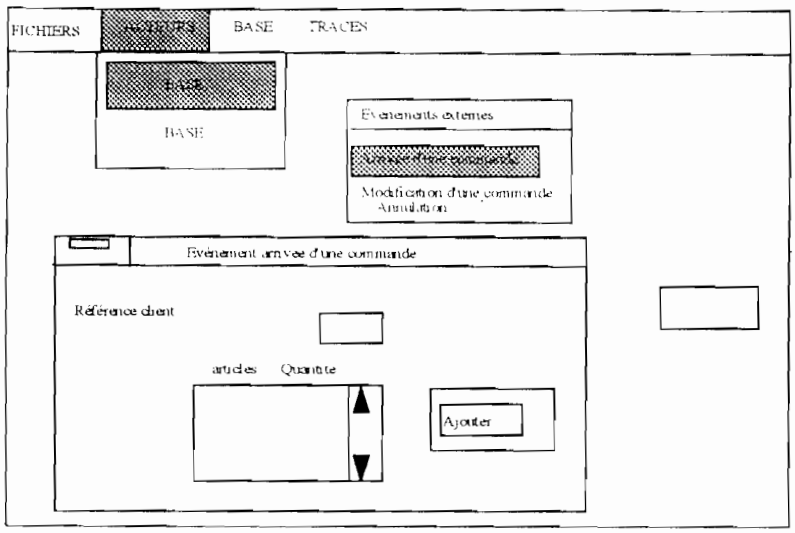

Figure 6.2 : Exemple d'interface d'un prototype

6.2. Passage des concepts O* en C++

Les modèles sémantiques de la méthode O* et du langage C++ [Ellis 90] s'appuient sur le paradigme objet ; mais la méthode O* utilise des concepts supplémentaires qui devront être simulés par la définition de mécanismes génériques écrits en C++.
Nous développons ci-après quelques règles d'implémentation de concepts O* en C++, bien que notre ambition soit de proposer un modèle pivot OO intermédiaire qui permettrait et faciliterait le passage du modèle O* vers différents langages OOs (Eiffel, C++, Smaltalk, SGBDOO, etc..).

la notion de domaine de valeurs
Si la sémantique du domaine de valeurs correspond à un type de donnée prédéfini du langage C++, nous adoptons ce type pour ce domaine de valeurs;
Exemple : les domaines ENTIER et RÉEL peuvent être rendus respectivement par les types long et double du C++..
Si elle correspond à la notion d'agrégation de domaines de valeur élémentaires, la notion de classe du langage C++ permet de les implémenter (données+opérations)
Exemple le domaine adresse devient la classe adresse ;
La notion d'ensemble de valeurs sera rendue par le concept de classe conteneur du langage C++.

Exemples : sacs, ensembles et listes

le lien de composition

La sémantique du lien de composition est bien rendue par la notion d'objet membre en C++ ; ainsi la création (resp. la destruction) d'un objet membre d'un autre objet se fait en même temps que la création (resp. la destruction) de l'objet lui même.

Si l'objet membre comporte des parties dynamiques, le langage prévoit leur création (resp. destruction) dans le constructeur (resp. destructeur) de l'objet.

Langage O*	Langage C++
classe PERSONNE	**classe** PERSONNE {
propriétés	CHAÎNE nom ;
nom : CHAÎNE	ENTIER âge ;
âge : ENTIER	//.....
	} ;
classe SUCCURSALES	**classe** SUCCURSALES {
propriétés	Ensemble<JourDeLaSemaine>
jours ouvrables :	JouersOuvrables ;
ensemble(JOUR DE LA SEMAINE)	//...
	};
classe OUVRAGE	**classe** OUVRAGE {
propriétés :	Liste<CHAINE> auteurs ;
auteurs : liste (CHAÎNE)	//.....
	} ;

N.B. Ensemble et Liste sont des classes C++ paramétrées par un type (classes template C++)

le lien de référence

Le lien de référence sera traduit par la syntaxe associée à un pointeur membre d'une classe (la classe référençante). On associera à l'objet référencé une liste de pointeurs vers les objets le référençant.

L'objectif visé ici est d'assurer la propriété d'inclusion des cycles de vie : le cycle de vie d'un objet référençant est inclus dans le cycle de vie de l'objet référencé ; un mécanisme de gestion d'exceptions placé dans le destructeur de l'objet référencé

permet d'autoriser ou non la désallocation de la mémoire de l'objet référencé selon que la liste des objets référençant est vide ou pas.

Lien d'héritage

La relation «est-un» entre classes O* peut être rendue par l'héritage entre classes C++. Dans C++, en effet, la création d'un objet d'une classe dérivée s'accompagne automatiquement de celle des parties d'objets des classes généralisées correspondantes. Il est possible de restreindre un objet au point de vue de n'importe quelle classe appartenant à sa hiérarchie d'héritage. Ceci nous permet de retrouver la sémantique de l'héritage O* qui se fonde sur le constat que les objets identifiés par l'analyste sont parfois des perspectives différentes d'un même phénomène du monde réel. Par exemple les objets "personne DIOUF" et "salarié DIOUF" ont conceptuellement les mêmes identités puisqu'ils sont tous les deux des vues particulières d'un même phénomène.

C++ offre un mécanisme de surcharge (dans une classe spécialisée d'opérations héritées de la classe) plus général que celui de O* autorisant les exceptions. On se restreindra en C++ au cas particulier d'ajout de caractéristiques supplémentaires (augmentation) : une fonction membre d'une classe dérivée qui surcharge (c'est à dire qui possède le même nom qu') une opération d'une classe de base contiendra en tête de son code un appel à la fonction de la classe de base. Ceci est possible en C++ en préfixant le nom de l'opération par l'opérateur de résolution de portée.

O* offre la possibilité de spécifier des contraintes sur les différentes instances des classes spécialisées

La contrainte de disjonction

Une contrainte de disjonction peut être définie entre classes spécialisées d'une même classe généralisée. L'intersection des instances des classes spécialisées est vide.

Exemple : **classe** VÉHICULE
assertions
disjonction (VOITURE,CAMION)

classe VOITURE **hérite de** VÉHICULE

classe CAMION **hérite de** VÉHICULE

Les classes CAMION et VOITURE héritent de la classe VÉHICULE avec une contrainte de disjonction. En C++, ce type de contrainte semble inutile puisqu'un objet instance de CAMION ne peut être instance de VOITURE. La contrainte de disjonction est traduite par un héritage classique.

La contrainte de couverture

La contrainte de couverture entre un ensemble de classes spécialisées exprime qu'à tout objet de la classe généralisée doit nécessairement correspondre au moins un objet appartenant à une des classes spécialisées.

Exemple:

classe PERSONNE
assertions
couverture(ÉTUDIANT, ENSEIGNANT)

classe ÉTUDIANT **hérite de** PERSONNE

classe ENSEIGNANT **hérite de** PERSONNE

On peut traduire cette contrainte en C++ par une classe PERSONNE abstraite (non instanciable), alors que ÉTUDIANT et ENSEIGNANT sont des classes concrètes persistantes. Pour tenir compte du fait que l'intersection n'est pas vide, nous pouvons être amenés à créer des classes de niveaux inférieurs en utilisant l'héritage multiple : on peut avoir des étudiants qui sont en même temps des enseignants.

La contrainte de partition

Une contrainte de partition entre un ensemble de classes spécialisées exprime à la fois une contrainte de disjonction et une contrainte de couverture. A tout objet de la classe généralisée doit nécessairement correspondre exactement un objet spécialisée appartenant à une des classes spécialisées ;

Exemple : **classe** PERSONNE
assertions
partition(HOMME, FEMME)

classe HOMME **hérite de** PERSONNE

classe FEMME **hérite de** PERSONNE

Les instances de PERSONNE sont soit des femmes, soit des hommes. Ils ne peuvent être que l'un ou l'autre (couverture) et mais pas les deux à la fois (disjonction). En C++ cette contrainte peut se traduire par une classe abstraite PERSONNE non-instanciable et deux classes FEMME et HOMME concrètes persistantes.

le graphe de transition d'état

Nous associons à chaque opération une méthode d'autorisation : BOLL OpérationEstApplicable().

Cette méthode, invoquée lors du déclenchement de l'opération, évalue l'assertion booléenne disjonctive dont les prédicats sont les classes d'états initiales de l'opération. L'opération n'est exécutée que si cette assertion (précondition) est vraie.

Exemple(Figure 6.3) : l'opération créer a comme précondition la classe d'états S0

l'opération 'ajouter' a comme précondition :

('rupture de stock' ou 'stock suffisant')

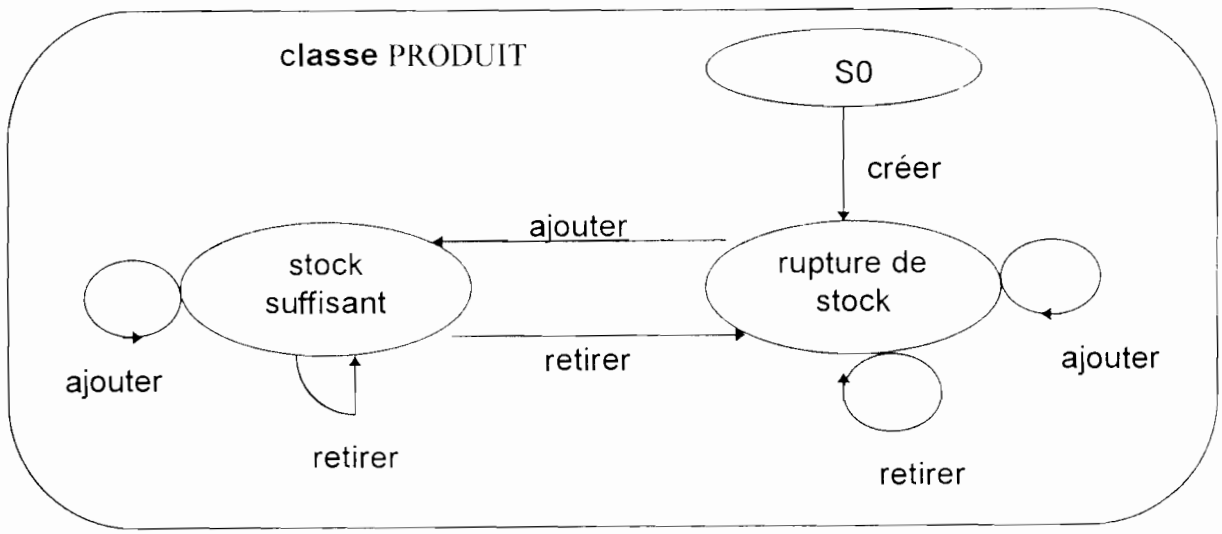

Figure 6.3 : Graphe de transition d 'état d'un produit

les contraintes d'attribut

Nous traduisons la contrainte d'attribut par des méthodes dites d'autorisation, appelées en tête des méthodes de création et de modification de la classe concernée.

Exemple

```
classe Salarie
{
char *NomSlarie;
int AgeSalarie;
// constructeur et destructeur
public:
Salarie(char*,int);
~Salarie();
// fonction d'autorisation
BOOL Salarie::b_ContrainteAttributSalarie();
};
BOOL Salarie::b_ContrainteAttributSalarie()
{
      if (AgeSalarie < 18) return FALSE;
      else return TRUE;
}
Salarie::Salarie(char*,int)
{
if (!b_ContrainteAttributSalarie())
~Salarie();
}
```

Le prototypeur sera capable de générer automatiquement les méthodes d'autorisation

6.3 La gestion des événements

Événements externes

La gestion des occurrences d'événements externes est faite grâce à une classe que nous faisons hériter d'une classe interface fenêtre (Sous Windows Borland C++ la classe TWindow - voir Figure 6.2) mettant à profit la capacité de cette dernière à réagir aux messages et événements de l'environnement de programmation. Ainsi la simulation de l'occurrence d'un événement externe consistera pour le développeur à sélectionner un item dans un menu. Ce menu possède la structure que nous avons donnée aux événements externes. La notion d'acteur de l'organisation a été utilisée comme critère de séparation dans l'ensemble des événements externes de l'application. Cet ensemble est partitionné en sous-ensembles, chacun a un rôle particulier joué par un acteur de l'organisation vis-à-vis du système d'information,.

Événements temporels

Nous devons implémenter un mécanisme de reconnaissance des événements temporels. Notre environnement nous permet de générer des "timers" pour envoyer à notre application des messages à intervalles de temps prédéterminés, simulant ainsi les occurrences des événements temporels.

Événements internes et cycle dynamique

La notion de cycle dynamique intègre (analogue à la notion de transaction atomique dans le vocabulaire des bases de données) est importante en O*. Elle correspond à la survenance d'une occurrence d'événement déclenchant un ensemble d'opérations qui font passer le SI d'un état cohérent à un autre état cohérent. Pour assurer une cohérence complète d'un SI, il faut prendre en compte les relations d'inférence entre événements : celles-ci n'ont de sens que si une occurrence d'événement inférée suit directement l'occurrence d'événement inférant correspondant.

Le gestionnaire d'événements

Le gestionnaire des événements temporels ainsi que le gestionnaire des événements externes permettent la simulation, par l'utilisateur, de ces types d'événements, dont les occurrences sont placées dans une FIFO en entrée du gestionnaire d'événements (Figure 6.4).

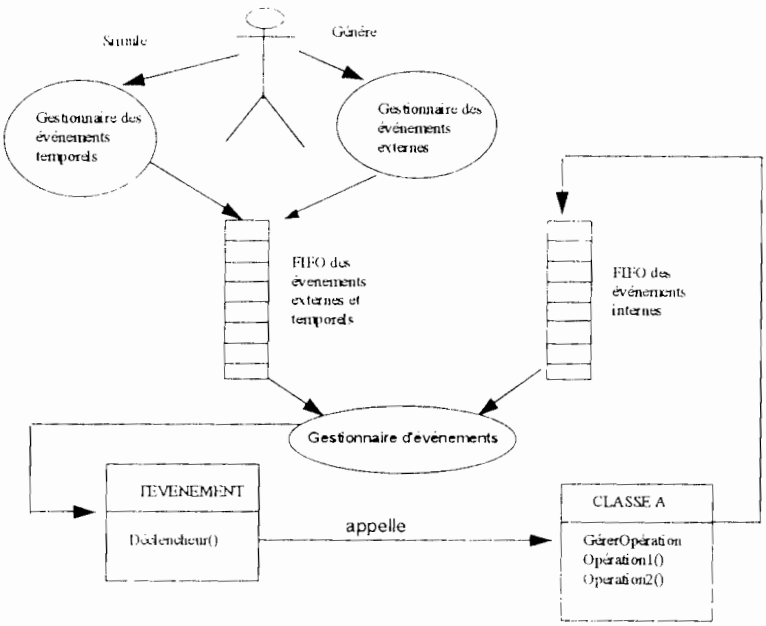

Figure 6.4 : le gestionnaire d'événements.

Pour chaque événement externe ou temporel, le gestionnaire d'événements appelle la fonction Déclencheur() de la classe TEVENEMENT. Cette fonction appelle la fonction GérerOpération() des classes concernées en spécifiant les opérations à déclencher.
La fonction GérerOpération() exécute les opérations demandées si c'est possible, puis identifie les événements internes valides de la classe et les placent dans la pile des événements internes. Le gestionnaire des événements applique ensuite le même traitement aux événements internes jusqu'à ce que la pile des événements soit vide; il reprend alors l'événement suivant de la pile des événements temporels et externes.
On remarquera que le gestionnaire des événements doit identifier les bouclages sur des événements internes lors de l'éxécution et proposer une éventuelle intervention.

6.4 Génération des squelettes de classes C++

Il s'agit produire à partir de la base de spécifications O* un ensemble de modules C++ sémantiquement équivalents, représentant la déclaration et la définition des classes C++ [Arial D / 95-2].
Tous les concepts vus comme des classes par le métamodèle de la méthode O* sont représentés au niveau interne grâce à des classes conteneurs. Ces classes conteneurs

sont disponibles par l'intermédiaire de la bibliothèque de classe fournie avec le compilateur C++. Nous présentons le traitement ci-après de quelque aspects statiques à savoir les classes, les liens d'héritage et les liens de référence:

Algorithme de traitement des classes

```
Déclarer un itérateur sur le dictionnaire des classes
Tant qu'on a pas rencontré la fin du dictionnaire des classes
    Début tant que
        Récupérer la classe pointée par l'itérateur
        Incrémenter l'itérateur
        Écrire ( 'class nom_classe') dans le fichier entête
        Traiter les liens d'héritage
        Traiter les liens de composition
        Traiter les liens de référence
        Traiter les propriétés à valeur dans domaine
    Fin Tant que
```

Algorithme de traitement des liens d'héritage

```
Début fonction
    Récupérer l'ensemble des liens d'héritage dont la classe passée en argument
        est origine
    Déclarer et initialiser un itérateur sur cet ensemble
    {Héritage simple}
    Si l'itérateur est non nul
     Début si
        Incrémenter l'itérateur
        si le lien existe
        Début si
            écrire(' : public nom_classe_extrémité_lien ')
        Fin si
     Fin si
    {Héritage Multiple}
    Tant que l'itérateur est non nul
    Début tant que
        Incrémenter l'itérateur
        si le lien existe
        Début si
            écrire(' , public nom_classe_extrémité_lien ')
        Fin si
     Fin si
    écrire('\n')
Fin fonction
```

Algorithme de traitement des liens référence

```
Début fonction
    Récupérer l'ensemble des liens de référence dont la classe passée en argument
      est origine
    Déclarer et initialiser un itérateur sur cet ensemble
    Tant que l'itérateur est non nul
     Début tant que
        Incrémenter l'itérateur
        si le lien existe
        Début si
            Récupérer le nom du lien de référence
            Récupérer le type du lien
            si c'est un lien simple
             début si
                écrire('nom_classe_extremite * nom_lien')
            sinon
                si c'est un lien multiple de type liste
                écrire('BAG nom_lien)
                si c'est un lien multiple de type ensemble
                écrire('SET nom_lien)
             Fin si
        Fin si
    Fin Tant que
Fin fonction
```

CONCLUSION

Nous avons fait la conception et parfois abouti à la réalisation d'un ensemble d'outils qui supportent le développeur dans le processus de création de système d'information :
-un éliciteur permettant la capture des besoins de l'utilisateur ;
-l'interface de saisie graphique qui comprend les 2 sous outils qui gèrent le schéma de classe et le graphe statique ;
-le gestionnaire de persistance qui étudie la possibilité de sauvegarde de manière automatique de la base en tables relationnelles sur disque dur à l'aide d'une interface C++/ SGBD ;
-le prototypeur qui génère du code C++ à partir de spécifications O* ;
Un élément important qui compléterait cet ensemble d'outils est le gestionnaire de processus lui même. Il a fait l'objet de travaux dans l'équipe de recherche [Grosz 91] [Rolland 93] [Rolland 94] [Clercin 95] [Balde 94].
Il reste un certain nombre de problèmes à résoudre dans la cadre de recherches futures, par exemple

-pour l'éliciteur
la description formelle des propriétés procédurales et non procédurables des actions de l'utilisateur et du système

-pour le prototypeur
la description par un langage formel des opérations des classes qui puisse être traduite de façon automatique en C++

- pour le gestionnaire de persistance et l'interface d'aide à la saisie de concept O*
la prise en compte des aspects dynamiques (opération, événement, graphes de transition d'état)

BIBLIOGRAPHIE

[Arial D / 95-1] Réalisation d'un compilateur O* à l'aide des outils Lex et Yacc, Rapport interne ARIAL, DAKAR, 1995.

[Arial D / 95-2] Conception et Réalisation d'un prototypeur pour O*, Rapport interne ARIAL, DAKAR, 1995.

[Arial D / 95-3] Gestion de la persistance dans un AGL supportant une méthode OO, Rapport interne ARIAL DAKAR, 1995.

[Bailin 89] S.C. Bailin, "An Object-Oriented Requirements Specification Method", Communications of the ACM, May 1989

[Baldé 94] Baldé S, Clercin Chr, De Vallois T, Sarr O. : "Conception et réalisation d'outils horizontaux complétant les méthodologies de systèmes d'information orientées objet ", in "Actes - Proceedings", CARI94, 1994.

[Baldé 96] Baldé S, Clercin Chr, Sarr O. : " Validation dans les outils de l'ingénierie des besoins orientée objet ", in " Actes - Proceedings ", CARI96, 1996.

[Boehm 84] B. W. Boehm: "Verifying and Validating Software Requirements and Design Specifications", Reprint IEEE Software Volume 1, Number 1, January 1984.

[Boehm 87] B. W. Boehm: "A Spiral Model of Software Development and Enhancement", Software Engineering Project Management, 1987 / reprinted in System and Software Requirements Engineering, IEEE Computer Society Press, Washington, DC , 1990

[Booch 82] G. Booch, "Object Oriented Design", Ada Letters, vol. 1, n°3. March/April 1982

[Booch 91] G. Booch, Object Oriented With Applications Benjamin Cumming Ed. 1991.

[Brodie 82] M.L. Brodie, E. Silva, " Active and Passive Component Modelling : A IFIP TC8 Int. Conference on Comparative Review of Information Sy Design Methodologies, North Holland, 1982

[Brunet 93] Brunet J. : "Analyse conceptuelle orientée-objet", thèse de doctorat de Paris VI, informatique, 1993.

[Caroll 96] J.M. Caroll Scenario-Based Design : Envisioning Work and Technology in System Development John Wiley & Sons,Inc 1996

[Castellani 91] X. Castellani, "Le modèle de la méthode MCO d'analyse et de conc systèmes d'objets", INFORSID, Paris, France, June 1991

[Cauvet 89] C. Cauvet, C. Rolland, C. Proix, "A design methodology for Object-Database", Int. Conference on Management of Data, Hyderabad, In

[Chen 76] P.P. Chen, " The entity relationship model : towards a unified view o ACM TODS, vol. 1, n°1, 1976

[Chen 92] Chen M., Norman R.J. : "A framework for integrated CASE", IEEE S vol.9, N°2, March 1992, pp. 18-22.

[Clercin 95] Chr. Clercin, Samba Baldé, Oumar Sarr, Thiérry De Vallois "Vérifica validation et prototypage dans un AGL supportant une méthode de spécification orientée objet" in "Actes du congrès INFORSID XIIIe c 30 Mai - 2 Juin 1995 Grenoble.

[Coad 90] P. Coad, E. Yourdon, Object Oriented Analysys, Second Edition Press, 1990.

[Costa 89] J.F. Costa, A. Sernadas, C. Sernadas "OBLOG User's Manual". Instituto Superior Técnico, Lisbon, May 1989

[DeMarco 79] T. DeMarco, Structured Analysis And System Specification. Yourdon Press, 1978

[Dubois 86] E. Dubois, J. Hagelstein,E. Lahou, F. Ponsaert, A. Rifaut, "A Knowledge Representation Language for Requirements Engineering", Proceedings of the IEEE, vol. 74, n°10,1986

[Dubois 94] Eric Dubois et al : Animating Formal Requirements Specifications of Cooperative Information Systems Proceedings of the Second International Conference on Cooperative Information Systems - CoopIS-94, Toronto(Canada), May 17- 20, 1994.

[Ellis 90] M.A. Ellis and B. Stroustrup, The Annoted C++ Reference Manuel, Addison-Wesley, 1990.

[Essink 91] L.J.B. Essink, W.J. Erhart, "Object Modelling and System Dynamics in the Conceptualization Stages of Information Systems Development", IFIP TC8/WG8.1 Working Conference on the Object-oriented Approach in Information Systems, Québec, Canada, Oct. 1991

[Fagan 86] M.E. Fagan . : "Advances in Software inspections" IEEE Trans. Softw. Eng. SE-12, 7 (July 1986) 744-75.

[Fagan 91] M.E. Fagan, J.C. Knight "Testing is not the best means of defect det and removal. Achieving Quality Software - A national debate", Socie Software Quality, San, Diego, Calif. (Jan. 1991)

[Fowler 91] M. Fowler, "The use of Object-Oriented Analysis in Medical Informat Large Integrated Systems", TOOLS 4, Prentice Hall, Paris, 1991

[Grosz 91] Grosz G., Rolland C. : "Computer Aided Requirements Engineering" European-Japonese Seminar on Information Modeling and knowledge Bases,Tokyo, May 1991.

[Grosz 94] Grosz G. : "A general framework for discribing the requirements engineering process", Int. Conf. on Systems, Man and Cybernetics, San Antonio, Texas. USA, Octobre 1994.

[Gustafsson 82] M. R. Gustafsson, T. Karlsson, J, A. Bubenko, «A Declarative Approach to Conceptual Information Systems Modeling », IFIP TC8 Int. Conferenceon Comparative Review of Information Systems Design Methodologies,North Holland, 1982

[Henderson 90] Henderson-Sellers B., Edwards J.M. : "The Object-Oriented System Life Cycle", Com. of the ACM, Sept 90, Vol 33, n°9, pp. 146-158.

[Henderson 91] Henderson-Sellers, "Analysis and Design, Methodologies and Notation", Tutorial, TOOLS, Paris, 1991

[Iivari 90] J. Iivari, "Hierarchical spiral model for information system and software development" Information and software technology, Feb. 1990

[Jacobson 92] Jacobson I., et al. Object-Oriented Sofware Engineering, A Use Case Driven Approach, Addisson Wesley, 1992.

[Junclaus 91] R. Jungclaus, G. Saake, T. Hartmann, C. Sernadas, "Object-Oriented Specification of Information Systems : The TROLL Language", Techn Universität Braunschweig, Dec. 1991

[Lemoigne 77] J.L. Lemoigne, La théorie du Ssystème Général, P.U.F, 1977

[Loomis 87] M.E.S. Loomis, A. V. Shah, J.E. Rumbaugh, "An Object Modeling Tec for Conceptual Design", European Conference on Object-Oriented Programming, Paris, June 1987

[Krief 91] Krief Ph. : "Utilisation des langages à objets pour le prototypage", Masson 1991, Paris.

[Meyer 90] Meyer B. : "Conception et programmation par objets. Pour du logiciel qualité", InterEditions, IIA, Paris, 1990.

[Mollaret 92] Mollaret Marc "Développer sous WINDOWS", Armand Colin, Paris mars 1992 pp.315--367.

[Moreno 93] Moreno M., Souveyet C. : "The Evolutionary Object Model (EOM)" in I Trans. A-30, (Ed.) Prakash et al. pp. 41-58.

[Nature 92] Jarke M., Sutcliffe A., Vassiliou Y., Rolland C., Bubenko J. : "Delivera NATURE.D-1, NATURE Esprit III project n° 6353", Dec 1992.

[Potts 94] Colin POTTS et al. : "Inquiry-Based Requirements Analysis", IEEE Software, 1994.

[Regnell 95] Björn Regnell et al Improving the Use case Driven Approach to Requirements Engineering IEEE Software 1995.

[Rolland 82] C. Rolland, C. Richard, " The Remora methodology for Information S Design and Management " , IFIP TC8 Int. Conference on Comparativ of Information Systems Design Methodologies, North Holland, 1982

[Rolland 86] Rolland C., Foucaut O., Benci G. : " Conception des systémes d'infor la méthode REMORA ", Eyrolles, Paris, 1986.

[Rolland 93] Rolland C. : "Modeling the requirements Engineering process", 3d E -Japanese Seminar on Information Modeling and Knowledge Bases, Budapest, Hungary, 1993.

[Rolland 94] Rolland C., Prakash Naveen : "Guiding the requirements engineering process" Asia-Pacific Software Engineering Conference, 1994.

[Rolland 97] C.Rolland and al A proposal for a scenario classification framework ESPRIT 4th Framework Programme CREWS 21.903. Cooperative Requirements Engineering With Scenarios 1997.

[Ross 77] D.T. Ross, K.G. Schoman, " Structured Analysis for Requirement Definition", IEEE Transactions on Software Engineering, vol. 3, n°1, January 1977

[Rumbaugh 91] Rumbaugh et al Object Oriented Modeling and Design Prentice Hall, Englewoods Cliffs, W.J. 1991.

[Rumbaugh 94] J. Rumbaugh "Getting Started. Using use cases to capture requirements" Journal of Object Oriented Programming", Sept 94.

[Schiel 89] U. Schiel, " OKAY - Object-Oriented Knowlegde Analysis and Design Inde, Dec. 1989

[Sernadas 89] A. Sernadas, J. Fiadero, C. Sernadas, H.D. Ehrich, "The Basic Buildi of Information Systems, Information Systems Concept", North Holland

[Shlaer 91] S. Shlaer, S. J. Mellor Object Lifecycles: Modeling the Word in States Prentice Hall, 1991.

[Smith 77] J.M. Smith, D.C.P. Smith, " Database Abstractions : Aggregation and Generalization" , ACM Transactions on Database Systems, vol. 2, n° June 1977

[Spaccapietra 89 S. Spaccapietra, C. Parent, "About Entities, Complex Objects and Object-oriented Data Models", Poceedings of the Working Conferenc on Information System Concepts, Namur, Belgium, October 1989

[Teisseire 91] M. Teisseire, P. Poncelet, A. Cavarero, S. Miranda, "A-HOOK, The object-oriented analysis of the HOOK system", report of External Eur Research Project, 1991

[Verheijen 82] G.M.A. Verheijen, J. Van Bekkum, " NIAM : an Information Analysis M IFIP TC8 Int. Conference on Comparative Review of Information Syst Design Methodologies, North Holland, 1982

[Vessey 92] Iris Vessey , Jarvenpaa S.L., Tractinsky N. : "Evaluation of vendor pr CASE tools as methodology companions" Com. of the ACM, Apr. 19 vol 35, n° 4, pp. 90-106.

[Wegner 89] P. Wegner, S. Zdonok, "Inheritance as an Incremental Modification or Like Is and Isn't Like", ECOOP, Oslo, Norway, Aug. 1988/ (shorter ve "Models of Inheritance", 2e Int. Workshop on Data Programming Lan Gleneden Beach, Oregon, June 1989

*ANNEXE 1: Métamodèle statique de O**

classe CONCEPT_O*
commentaires
Cest la classe (abstraite) racine du métamodèle
propriétés
nom_CONCEPTO* : STRING
assertions
partition(ASSERTION, CLASSE, LIEN,EVENEMENT, G.T.E.,
CLASSE_D_ETAT, PROPRIETE, OPERATION,DOMAINE,
COMPOSANT_DOMAINE)

classe LIEN hérite de CONCEPT_O*
commentaires
C'est la classe qui exprime en général des relations entre diverses classes. Les propriétés vont être précisées par des assertions dans les classes spécialisées de LIEN
références
Origine_du_LIEN : CONCEPT_O*
Extrémité_du_lien : CONCEPT_O*
assertions
partition(LIEN_STATIQUE,LIEN_DYNAMIQUE)

classe CLASSE hérite de CONCEPT_O*
commentaires
Objet principal du modèle. Les occurrences de la classe CLASSE sont les classses de l'application cible
propriétés
nom_de_CLASSE : STRING
opérations_de_classe : set_ of (OPERATION)
GTE_de_CLASSE : opt set_of (GTE)
asssertions_de_CLASSE : opt set of (ASSERTION)
assertions
unique(nom_de_CLASSE)

classe LIEN_STATIQUE hérite de LIEN
assertions
partition(LIEN_STATIQUE_d_ATTRIBUT,
LIEN_STATIQUE_DE_CLASSE)
Origine_du_LIEN : CLASSE

classe LIEN_STATIQUE_d_ATTRIBUT hérite de LIEN_STATIQUE
commentaires
C'est une assertion qui fait la spécialisation
assertions
Extrémité_du_LIEN : PROPRIETE

classe LIEN_STATIQUE_DE_CLASSE hérite de LIEN_STATIQUE
commentaires
c'est une assertion qui fait la spécialisation
assertions
partition(LIEN_DE_COMPOSITION_DE_CLASSE,
LIEN_DE_REFERENCE,
LIEN_D_HERITAGE)
Extrémité_du_LIEN : CLASSE

classe LIEN_COMPOSITION_DE_CLASSE hérite de LIEN_STATIQUE_DE_CLASSE
commentaires
valuation_LCC pouvait être typé par BOOLEEN
plutôt que par énumération
propriétés
nom_de_LCC : STRING
valuation_LCC : {simple, multiple}
assertions
unique dans composé(nom_de_LCC)

classe LIEN_DE_REFERENCE hérite de LIEN_STATIQUE_DE_CLASSE
commentaires
valuation_LR pouvait être typé par BOOLEEN plutôt que par énumération
propriétés
nom_de_LIEN_REFERENCE : STRING
valuation_LR :{simple,multiple}
assertions
unique dans omposé(nom_de_LIEN_REFERENCE)

classe LIEN_DE_HERITAGE hérite de LIEN_STATIQUE_DE_CLASSE
commentaires
L'origine du lien c'est la classe spécialisée,
l'extrémité la classe généralisé

classe LIEN_DYNAMIQUE hérite de LIEN
commentaires
La spécialisation consiste en une asserion de partition
assertions
partition(DECLENCHEUR,
TRANSITION_D_ETAT)

classe DECLENCHEUR hérite de LIEN_DYNAMIQUE
commentaires
Le déclencheur est devenu une classe alors qu'il n'en est pas une dans le modèle O* d'origine.Cette classe apporte une simplification intéressante à savoir une valuation « simple » de tous les liens
propriétés
facteur_pour : opt STRING
condition_si : opt STRING
assertions
origine_du_LIEN : EVENEMENT
extrémité_du_LIEN : OPERATION

classe EVENEMENT hérite de CONCEPT_O*
commentaires
Cette classe constitue dans le modèle d'origine un lien dynamique spécialisé. L'ajout de DECLENCHEUR modifie cette hiérarchie;la classe EVENEMENT est composée dans CLASSE ; DECLENCHEUR n'est pas composée dans EVENEMENT pour l'AGL
propriétés
nom_d_EVENEMENT : STRING
assertions
partition(EVENEMENT_INTERNE, EVENEMENT_EXTERNE, EVENEMENT_TEMPOREL)
unique dans composé(nom-d-EVENEMENT)

classe EVENEMENT_INTERNE hérite de EVENEMENT
propriétés
prédicat : opt STRING

classe EVENEMENT_EXTERNE hérite de EVENEMENT
propriétés
message : opt STRING

classe EVENEMENT_TEMPOREL hérite de EVENEMENT
commentaires
Malgré la ressemblance de forme avec événement interne, nous conservons cette spécialisation du fait de la sémantique différente de l'événement interne
propriétés
nom_PREDICAT_TEMPOREL : opt STRING

classe GTE hérite de CONCEPT_O*
propriétés
nom_de_GTE : opt STRING
nom_de_CLASSE_ETAT : set of (CLASSE_d_ETAT)
références
énumération_TE : set of (TRANSITION_D_ETAT)
assertions
unique dans composé(nom_de_GTE)

classe TRANSITION_D_ETAT hérite de LIEN DYNAMIQUE
références
effectuée_par : OPERATION
assertions
Origine-du_LIEN : CLASSE_D_ETAT
Extrémité du LIEN : CLASSE_D_ETAT

classe PROPRIETE hérite de CONCEPT_O*
commentaires
Cette classe a pour occurrences les noms des propriétés de l'application cible
propriétés
nom_de_propriété : STRING
références
valeur_dans : opt DOMAINE
assertions
unique dans composé (nom_de_PROPRIETE)

classe OPERATION hérite de CONCEPT_O*
commentaires
Cette classe a pour occurrences les noms des opérations de l'application cible
propriétés
nom_d_OPERATION : STRING
spécification_d'opération : opt STRING
références
propriétés_concernées : opt PROPRIETES
assertions
unique dans composé (nom_d_OPERATION)

classe ASSERTION hérite de CONCEPT_O*
propriétés
texte_d_ASSERTION: opt STRING
références
opérande : set_of (LIEN_STATIQUE)
assertions
partition(CONTRAINTE_D_ATTRIBUT,
CONTRAINTE_D_UNICITE,
CONTRAINTE_D_HERITAGE)
opérande : limité aux liens statiques dont l'origine est la classe possédant l'assertion

classe CONTRAINTE_D_ATTRIBUT hérite de ASSERTION
commentaires
Un opérande peut être tout saus un lien d'héritage
Ceci est bien rendu par l'assertion suivante
assertions
opérande : non (LIEN_D_HERITAGE)

classe CLASSE_D_ETAT hérite de CONTRAINTE_D_ATTRIBUT
commentaires
Cette classe a pour occurences les noms des
ETATS dans lesquels peut passer toute instance
d'une classe.
propriétés
nom_de_CLASSE_D_ETAT : {STRING,CLASSE}
assertions
partition(CLASSE_D_ETAT_CORRELEE,
CLASSE_D_ETAT_CALCULEE)

classe CLASSE_D_ETAT_CALCULEE hérite de CLASSE_D_ETAT
commentaires
Cette spécialisation se justifie pour récupérer la connaissance des liens statiques à découvrir dans la contrainte d'attribut. En effet une assertion de classe d'état est identique à une assertion contrainte d'attribut
assertions
(nom_de_CLASSE_D_ETAT) valorisée en partie
droite par STRING

classe CLASSE_D_ETAT_CORRELEE hérite de CLASSE_D_ETAT
commentaires
correspond à la partie droite CLASSE. La classe doit
être une classe spécialisée
assertions
(nom_de_CLASSE_D_ETAT) valorisée en partie droite par CLASSE
L'occurrence de CLASSE doit être une classe spécialisée

classe CONTRAINTE_D_UNICITE hérite de ASSERTION
propriétés
opérateur : {unique dans composé,unique}

classe CONTRAINTE_D_HERITAGE hérite de ASSERTION
propriétés
opérateur : {partition,couverture,disjonction}
assertions
opérandes limités aux classes spécialisées

classe DOMAINE hérite de CONCEPT_O*
commentaires
Cette classe a pour occurrences les noms des
domaines existant et/ou crées par le développeur
assertions
partition(DOMAINE_SIMPLE,DOMAINE_COMPLEXE)

classe DOMAINE_SIMPLE hérite de DOMAINE
commentaires
le domaine simple spécialise le domaine
assertions
partition(DOMAINE_PREDEFINI, DOMAINE_ENUMERE)

classe DOMAINE_PREDEFINI hérite de DOMAINE_SIMPLE
commentaires
Cette classe est le domaine élémentaire
propriétés
nom_de_domaine : opt STRING
type_domaine : {STRING,INT,REAL,DATE,BOOL, autre}
assertions
il existe nom_de_domaine si type domaine =« autre »
unique dans composé(nom_de_DOMAINE)

classe DOMAINE_ENUMERE hérite de DOMAINE_SIMPLE
commentaires
Cette classe contient l'énumération de noms de domaines simples
propriétés
valeurs_domaines : set_of (STRING)

classe DOMAINE_COMPLEXE hérite de DOMAINE
assertions
partition (DOMAINE_AGREGAT, DOMAINE_COLLECTION)

classe DOMAINE_AGREGAT hérite de DOMAINE_COMPLEXE
commentaires
la classe la plus complexe
propriétés
composants_de_domaine : set_of (COMPOSANT_DOM_AGR)

classe COMPOSANT_DOMAINE_AGREGAT hérite de CONCEPT_O*
propriétés
libellé_CDA : STRING
références
domaine_agrégé : DOMAINE
assertions
unique dans composé (libellé_CDA)

classe DOMAINE_COLLECTION hérite de DOMAINE_COMPLEXE
commentaires
aussi complexe que Domaine agrégat
propriétés
libellé_DC : STRING
références
domaine_membre : DOMAINE

ANNEXE 2: Code implémentant l'interface de saisie

(menu général, graphe statique, schéma local de classe)

Le code a été réparti suivant plusieurs modules (fichiers), selon les fonctionnalités qu'ils sont censés gérer. La déclaration (l'interface) des classes d'un module donné se trouve dans un fichier d'extension .hpp et l'implémentation des fonctions membres déclarés dans le fichier d'extension .hpp précédent, se trouve dans un fichier d'extension.cpp.

Entête ou interface	Implémentation	Fonctions gérées
menug.hpp	menug.cpp	le menu général (ouvrir une base, la fermer, créer une nouvelle base, la quitter le passage vers un schéma de classe, vers le graphe statique
con_tda.hpp	con_tda.cpp	les classes conteneurs et leurs dérivés : le dictionnaire de classe, les conteneurs des différents liens statiques
bdialtxt.hpp	bdialtxt.cpp	les boites de dialogues
interfac.hpp	interfac.cpp	les fenêtres associées aux schémas de classe et au graphe statique, les fonctions de réponse aux messages en provenance de l'environnement Windows
objets.hpp	objets.cpp	les objets conceptuels : le schéma de classe, le graphe statique, propriétés, assertions opérations....
paletgst.hpp	paletgst.cpp	la barre d'outils de la fenêtre associée au graphe statique
liens.hpp	liens.cpp	les liens conceptuels
liensgr.hpp	liensgr.cpp	la représentation graphique des liens conceptuels sur les différentes fenêtres
flots.hpp	flots.cpp	les flots fichiers
aideg.hpp	aideg.cpp	l'aide en ligne
menubor.rc	menubor.res	les ressources (menus, images, boites de dialogue, formes du curseur).
prototyp.hpp	prototyp.cpp	générations des squelettes de classes

Pour des raisons de place nous n'avons pu inclure dans ce mémoire que les fichiers suivants : menug.hpp, menug.cpp, con_tda.hpp et con_tda.cpp

```
//============================================================================
//                NOM DE FICHIER: MENUG.HPP
//
//
//Liste des classes:
//                .Classes logicielles
//                     -TApp
//                     -Programme principal
//                     -
//                .Classes interfaces
//                                -TFenPrinc
//
//============================================================================

//*********REMARQUES:

//*********Indications compilateur
#ifndef  WIN30
#define WIN30
#endif

//*********Liste des fichiers inclus
#ifndef _PROTO_HPP
#define _PROTO_HPP

#include <owl.h>
#include <listbox.h>
#include <dialog.h>
#include <static.h>
#include <edit.h>
#include "proto.h"
#include "myclstyp.h"
#include <filedial.h>
#include <listbox.h>
#include <stdio.h>
#include "bdialtxt.hpp"
#include <windobj.h>

//*********Declaration des classes externes utilisees
class TVerification ;
class TGrapheStatique ;

//****************************************************************************
//****************************************************************************
//                NOM DE LA CLASSE: TApp
//
//Description:
//
//Parametres de creation:
//        Nom                     : nom de l'application
//        hInst                   : handle de l'application (instance courante)
//        hPrevInst               : handle de l'application précédente
//        lpCmdLine               : pointeur vers la ligne de commande
//        hCmdShow                : taille requise pour la fenêtre principale au démarrage
//
//Comportement du destructeur:
//
//Liste des ancetres: TApplication
//
//Liste des enfants:
```

```
//
//Classes amies:
//
//Operateurs surcharges:
//------------------------------------------------------------------------

//**************************************************************************
//      CLASSE                          : TApp  - Déclaration de la classe
//      DESCRIPTION             : c'est la classe Application de notre logiciel Outil
//
//      PARAMETRES DE CREATION ET REMARQUES

//
//      FONCTIONS MEMBRES PUBLIQUES
//      TApp                    : le constructeur
//      InitMainWindow          :
//
//      SUPERCLASSES            :
//      CLASSES DERIVEES: Aucune
//      CLASSE AMIES            : Aucune
//**************************************************************************
_CLASSDEF(TApp)
class TApp : public TApplication
{
//                              ++++++++++++++++++++PUBLIQUE+++++++++++++++
public :

//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
        TApp(LPSTR Nom, HANDLE hInst, HANDLE hPrevInst,
        LPSTR lpCmdLine, int nCmdShow) : TApplication(Nom, hInst,
        hPrevInst, lpCmdLine, nCmdShow)      {}

//->[]Methodes de manipulation
        virtual void InitMainWindow() ;
//                          crée un objet fenêtre principale

        virtual void InitInstance()
        {
        TApplication::InitInstance();
        HAccTable =
          LoadAccelerators(hInstance, "FILECOMMANDS");
        }


} ;


//**************************************************************************
//**************************************************************************
//              NOM DE LA CLASSE: TFenPrinc
//
//Description:
//
//Parametres de creation:
//      LPSTR                   . nom de la fenêtre
//
//Comportement du destructeur:
//
//Liste des ancetres: TWindow
```

```
//
//Liste des enfants:
//
//Classes amies: TSchemaClasse
//
//Operateurs surcharges:
//------------------------------------------------------------------------

_CLASSDEF(TFenPrinc)
class TFenPrinc : public TWindow
{

        friend TSchemaClasse ;
        friend TFenSchemaClasse ;

//                                  ------------------PRIVE----------------
//@ @ @ @ @ @ @ @ @ @  Liste des donnees
//réponse à la boite de dialogue affichée lors de la femeture d'une
//base pour la sauvegarde ou non des modifications (IDYES,IDNO,IDCANCEL)
//cette variable sert pour par exemple Trt_Ouvrir qui envoie un message
//de fermeture; elle doit savoir si le développeur a choisi IDYES,IDOK
//et IDCANCEL
        int                             SaveBeforeClose ;
        BOOL  SchemaEstOuvert ;
                                                //indique si une base est ouverte
                                //item ouvrir ou nouveau
        BOOL  bSchemaClasse, // on a cliqué sur classe
                        bGrapheStatique, // on a cliqué sur statique
                        bGrapheDynamique ,// on acliqué sur dynamique
                        bAide ; // booléen d'aide demandée

        //les menus fottants
        HMENU hMenuClasse, // clic sur un nom de classe
        hMenuGraphDyn ; // clic sur un nom de graphe dynamique
        void  KillAllChildren()
         { ForEach( (TActionMemFunc) (&TFenPrinc::KillAChild) , NULL ); }
  void   KillAChild(Pvoid , Pvoid ) ;

 void  RafraichirUneFenFille(Pvoid , Pvoid ) ;


//{{{{{{{{{{ Methodes de reponses a un evenement
//fonction utilisées pour l'aide en ligne
        void                            WMEntrerInactif(RTMessage Msg)
                                                = [WM_FIRST + WM_ENTERIDLE] ;

//<< >> << >> Methode de gestion des flux
        virtual const Pchar streamableName() const ;
        virtual Pvoid           read(Ipstream ) ;
        virtual void            write(Opstream ) ;
        TFenPrinc(StreamableInit) ;
        void OpenFileExistantEnEntree() ;
        void OpenFileInExistantEnSortie() ;

//                              ++++++++++++++++++PUBLIQUE++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees

        TGrapheStatique * GrapheStatique ;
        TVerification    * Verification ;
```

```
        TListBox        * Liste,
                                        * RightBoiteListe;
        static unsigned int AppelsFenGrStDispAll ;

        // nom de la base courante (chemin inclus )
        char                    NomFichier[MAXPATH] ;
        //lecteur
        char        drive[MAXDRIVE] ;
    //chemin
        char        dir [MAXDIR] ,
    //nom fichier (8 car. maxi)
        char        file[MAXFILE] ;
        //extension du fichier
        char        ext[MAXEXT] ;
        //valeur de retour de la fonction fnsplit;5 bits permet de savoir
        //les composants du chemin DIRECTORY, DRIVE, EXTENSION, FILENAME,
    //WILDCARS
        int         flags ;

        BOOL                    IsDirty,IsNewFile ;
                        //isDirty indique des modifications
                        // isNewFile pas de nom de fichier connu pour la base

        int                             IndexClasse, // mémorise l'index de l' item correspondant
                                        IndexGraphDyn ; // à un nom de classe (ou de graphe
dynamique)
                                                                // donné (pour les fonctions
                                                        // réponse aux items des menus
flottants
        char                    NomClasse[80] ;
        char                    NomClassePRED[80] ;
        char                    NomGraphDyn[80] ;
        char                    szNomClasseCree[80] ;
        //pour la boite de dialogue Fait le
        char                    szFait_le[80] ;
//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
        TFenPrinc(LPSTR ) ;
        ~TFenPrinc();

//[]->Methodes d'information
        virtual LPSTR   GetClassName () ;//def. icone appli

//->[]Methodes de manipulation

        void   SplitCheminFichier(LPSTR );

        virtual void        SetupWindow() ;

        virtual   BOOL  CanClose() ;

        void     DeleteAll();
//                      Initialise toutes les variables

//envoie un message PAINT  à toutes les fenetres
void  RafraichirToutesLesFenFilles()
  {
        ForEach
        (
         (TActionMemFunc) (&TFenPrinc::RafraichirUneFenFille) , NULL
```

```
        );
    }

//lors d'une modification concernant une classe(par exemple renommage
//mettre à jour le contenu graphique des fenêtres qui
//mentionnent cette classe dans leur zone client ; elle possède
// comme argument un pointeur vers un TSchemaClasse
void MajFensConcerneParModifClass(TSchemaClasse *) ;
//lors d'une suppression concernant une classe
//mettre à jour le contenu graphique des fenêtres qui
//mentionnent cette classe dans leur zone client ; elle possède
// comme argument TListeDeLiens*
void MajFensConcerneParSupprClass(TListeDeLiens*) ;


        virtual void      GetWindowClass (WNDCLASS &) ;
//                        def. icone appli

//<< >> << >> Methode de gestion des flux
        virtual void      OpenFile() ;
//                        appeler par Trt_Ouvrir

        virtual void      SaveFile() ;
//                        appeler par Trt_Enregistrer

        virtual int     SaveFileAs() ;
//                        appeler par Trt_Enregistrer sous

          void    SetIsDirty(BOOL bValue) { IsDirty = bValue ; }
//                        appelé par TDicoClasses::EstMembre(LPSTR,TSchemaClasse* &,
//          BOOL & ,HWND )

void DeleteStrngInRightLstBox( int ) ;
//      appelée par TDicoClasses::DetruireClasseEtDependances

        static    PTStreamable build() ;

//{{{{{{{{{{{ Methodes de reponses a un evenement
        //~~~ messages en provenance des menus WM_COMMAND
        void                    Trt_Ouvrir(RTMessage Msg)
                                        = [CM_FIRST+IDM_OUVRIR] ;

        void                    Trt_Quitter(RTMessage Msg)
                                        = [CM_FIRST+IDM_QUITTER ] ;

        void                    Trt_Nouveau(RTMessage Msg)
                                        = [CM_FIRST+IDM_NOUVEAU] ;

        void                    Trt_Enregistrer(RTMessage Msg)
                                        = [CM_FIRST+ IDM_ENR] ;

        void                    Trt_EnregistrerSous(RTMessage Msg)
                                        = [CM_FIRST+IDM_ENRSOUS] ;
        void                    Trt_FermerBase(RTMessage Msg)
                                        = [CM_FIRST+IDM_FERMERBASE] ;

        void Trt_Cplusplus(RTMessage) = [CM_FIRST+IDM_CPP] ;
        /*
        void Trt_SpTextuelles(RTMessage) = [CM_FIRST+IDM_TEXT] ;
```

```
        void Trt_Cplusplus(RTMessage) = [CM_FIRST+IDM_CPP] ;
        void Trt_SpEiffel(RTMessage) = [CM_FIRST+IDM_EIFFEL] ;
        void Trt_SpPCTE(RTMessage) = [CM_FIRST+IDM_PCTE] ;
        void Trt_ControleValidite(RTMessage) = [CM_FIRST+IDM_VALID] ;

        void Trt_ImpGraph(RTMessage) = [CM_FIRST+IDM_PRINT] ;
        void Trt_ListeDomaine(RTMessage) = [CM_FIRST+IDM_LISTEDOM] ;
        void Trt_Edit(RTMessage) = [CM_FIRST+IDM_EDIT] ;
        void Trt_Options(RTMessage) = [CM_FIRST+IDM_OPTIONS] ;
        */

        //~~~~ messages en provenance d'un ID enfant
        void                    Trt_Liste(RTMessage Msg)
                                        = [ID_FIRST + ID_Liste] ;
//                              message envoyer par la 2eme boite liste de (noms)

        void                    Trt_SchemaStaticDynamic(RTMessage Msg)
                                        = [ID_FIRST + ID_RightBoiteListe] ;
//                              message envoyer par la première boite liste


        //~~~~ menu flottant suite à un clic sur un nom de classe
        void                    RenommerClasse(RTMessage Msg)
                                        = [CM_FIRST + IDM_RENCLASSE] ;
        void                    SupprimerClasse(RTMessage Msg)
                                        = [CM_FIRST + IDM_DROPCLASSE] ;
        void                    EditerSchemaClasse(RTMessage Msg)
                                        = [CM_FIRST + IDM_EDITCLASSE] ;

        //~~~~ menu flottant suite à un clic sur un nom de graphe dynamique
        void                    RenommerGrapheDynamique(RTMessage Msg)
                                        = [CM_FIRST + IDM_RENGRAPHDYN] ;
        void                    SupprimerGrapheDynamique(RTMessage Msg)
                                        = [CM_FIRST + IDM_DROPGRAPHDYN] ;

        virtual void    CMAbout(RTMessage Msg)
                                        = [CM_FIRST+ IDM_ABOUT] ;
        void                    CMIndex (RTMessage Msg)
                                        = [CM_FIRST+ IDM_INDEX] ;


//      void EditerGrapheDynamique(RTMessage) = [CM_FIRST + IDM_EDITGRAPHDYN] ;

} ;


#endif

//============================================================================
//              NOM DE FICHIER: MENUG.CPP
//
//
//Liste des classes:
//          -TFenPrinc
//                              -TVerification
//                              -TApp
//
//Remarques:
//Ce fichier contient une partie du code de definition des operateurs
//de FLUX et la définition de la boîte liste de droite
```

```
//=============================================================================

//*********Liste des fichiers inclus
#include <assoc.h>
#include <strng.h>
#include <dict.h>
#include <string.h>
#include <stdio.h>
#include <array.h>
#include "con_tda.hpp"
#include <dos.h>
#include <process.h>
#include "menug.hpp"
#include "objets.hpp"
#include "verific.hpp"
#include "interfac.hpp"

//=*=*=*=*=*=*=*=*=*=*VARIABLES GLOBALES=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=
TDicoClasses          * DicoClasses ;
TConteneurLiensHERIT   * ConteneurLiensHERIT;
TConteneurLiensCOMPOS  * ConteneurLiensCOMPOS ;
TConteneurLiensREFER   * ConteneurLiensREFER ;
TTableauProprietesDsDom * ConteneurProprietes;
TTableauDomaines       * ConteneurDomaines ;
//*****************************************************************************
//*****************************************************************************
//              METHODES DE LA CLASSE: TFenPrinc
//*****************************************************************************

//++++++++-------- CONSTRUCTEURS ET DESTRUCTEUR
//-------------
//Parametres:
//       -LPSTR: tittre de la fenetre
//Renvoie:
//Donnees de la classe utilisee.
//       -Attr.Style(M): def. des caracteristiques de la fenetre
//Objets crees:
//       -menug, 2 listbox(choix modele et graphes existants)
//       -TVerification
//       -menu flottant hMenuGraphDyn, hMenuclasse (rename,edit,suppr)
//Commentaires:
//
//
const int VALEURPARDEFAUT = 10 ;//pour TFenPrinc::Trt_Ouvrir
TFenPrinc::TFenPrinc(LPSTR Titre) : TWindow(NULL, Titre)
{//début fonction

Verification = new TVerification() ;

Attr.Style |= WS_CLIPCHILDREN ;
AssignMenu("menug") ;

Liste = new TListBox(this, ID_Liste,50,30,220,270) ;

Liste->Attr.Style &= ~LBS_SORT ; // liste non tiée
Liste->Attr.Style |= WS_CAPTION | WS_SIZEBOX | WS_CLIPCHILDREN ;
Liste->Attr.Style &= ~WS_MAXIMIZEBOX ; // lui oter sa case d'agrandissement
Liste->Attr.Style &= ~WS_VISIBLE ; // invisible au départ
Liste->Title = "Vues..." ;
```

```
RightBoiteListe = new TListBox(this,ID_RightBoiteListe,350,30,220,270) ;
RightBoiteListe->Attr.Style |= WS_CAPTION | WS_SIZEBOX |WS_CLIPCHILDREN ;
RightBoiteListe->Attr.Style &= ~WS_MAXIMIZEBOX ; // lui oter sa case d'agrandissement
RightBoiteListe->Attr.Style &= ~WS_VISIBLE; // invisible au départ

// menu flottant suit à un clic sur un nom de classe
hMenuClasse = CreatePopupMenu() ;
AppendMenu(hMenuClasse, MF_STRING, IDM_RENCLASSE, "Renommer") ;
AppendMenu(hMenuClasse, MF_STRING, IDM_DROPCLASSE, "Supprimer") ;
AppendMenu(hMenuClasse, MF_STRING, IDM_EDITCLASSE, "Editer") ;

// menu flottant suite à un clic sur un nom de graphe dynamique
hMenuGraphDyn = CreatePopupMenu() ;
AppendMenu(hMenuGraphDyn, MF_STRING, IDM_RENGRAPHDYN, "Renommer") ;
AppendMenu(hMenuGraphDyn, MF_STRING, IDM_DROPGRAPHDYN, "Supprimer") ;
AppendMenu(hMenuGraphDyn, MF_STRING, IDM_EDITGRAPHDYN, "Editer") ;
DeleteAll() ;
}//fin fonction
//-------------
//                          DESTRUCTEUR
//Parametres:
//Renvoie:
//Donnees de la classe utilisee:
//Objets crees:
//Commentaires:
//
//
TFenPrinc::~TFenPrinc()
{//début fonction
 //delete  Liste ;
 //delete  RightBoiteListe  ;
 delete  Verification ;
 delete  GrapheStatique ;
 delete  ConteneurLiensHERIT,
 delete  ConteneurLiensCOMPOS ,
 delete  ConteneurLiensREFER
 delete  DicoClasses ;
 delete  ConteneurProprietes;
 delete  ConteneurDomaines;
}//fin fonction
//->[] Methodes de manipulation
//-------------
//        NOM DE LA METHODE: DeleteAll()
//
//Parametres:
//Renvoie:
//Donnees utilisees                                (M) indique une modif de la donnee
//        -cf code
//Methodes appelees:
//
//Commentaires:
//
void TFenPrinc::DeleteAll()
{
GrapheStatique = NULL ;
DicoClasses    = NULL ;
ConteneurLiensHERIT=  NULL ;
ConteneurLiensCOMPOS = NULL ;
ConteneurLiensREFER = NULL ;
```

```
ConteneurProprietes= NULL;
ConteneurDomaines=NULL ;
SchemaEstOuvert = FALSE ;
bSchemaClasse = bGrapheStatique = bGrapheDynamique = FALSE ;
IndexClasse = IndexGraphDyn = 0; //par précaution
NomClasse[0] = 0 ;
//NomFichier[0] = 0 ;
IsNewFile = TRUE ;
IsDirty = FALSE ;

SaveBeforeClose = VALEURPARDEFAUT;


}



//<< >> << >> << >>Methode de gestion des flux
//-------------
//        NOM DE LA METHODE: OpenFile()
//
//Parametres:
//Renvoie:
//Donnees de la classe utilisee: NomFichier, IsNewFile(M), Isdirty(M), SchemaEstOuvert(M)
//
//Commentaires:
//on force la fenetre à se redessiner apres la lecture du fichier
//
void TFenPrinc::OpenFile()
{//début fonction

//Cette fonction est appelée par Trt_Ouvrir qui a récupéré un nom
//de base (chemin inclus) à l'aide de la boite de dialogue SD_FILEOPEN
//ici on regarde si ce nom (introduit) par le développeur correspond
//à un nom de base existante auquel cas il faut ouvrir cette base
//en lecture
//sinon ouvrir la base écriture voir else de if (!done)
struct find_t ffblk ;
int done = _dos_findfirst(NomFichier,_A_NORMAL, &ffblk);
if(!done)
  {//début if (!done)

        OpenFileExistantEnEntree();

  }//fin if(!done)
else
  {//début else if(!done)

        OpenFileInExistantEnSortie() ;

  }//fin else if(!done)


}//fin fonction

//<< >> << >> << >>Methode de gestion des flux
//-------------
//        NOM DE LA METHODE:OpenFileExistantEnEntree
//
//Parametres:
```

```
//Renvoie:
//Donnees de la classe utilisee: NomFichier, IsNewFile(M), Isdirty(M), SchemaEstOuvert(M)
//
//Commentaires:
//on force la fenetre à se redessiner apres la lecture du fichier
//


void TFenPrinc::OpenFileExistantEnEntree()
{//début fonction

        ifpstream
                is (NomFichier ): // ouvre un flux en entrée

        char  bufifp[180] ; strcpy(bufifp,"Impossible d'ouvrir le fichier ");
        strcat(bufifp,NomFichier) ; strcat(bufifp," en lecture !!");
        if( is.bad() )
                MessageBox(HWindow,bufifp,
                 "O Etoile - Erreur Fichier",MB_OK|MB_ICONEXCLAMATION) .
   else
        {//début else is.bad()
           DeleteAll() ;
           read(is) ;
           is.close() ;
           //afficher la boite liste de gauche
           ShowWindow(Liste->HWindow, SW_SHOWNORMAL) ;
           //afficher le nom du fichier en haut de la fenetre principale
           char buftitle[300]; strcpy(buftitle, "O Etoile - ");
           strcat(buftitle,NomFichier);
           SetCaption(buftitle) ;
           IsNewFile = IsDirty = FALSE ;
        SchemaEstOuvert = TRUE ;
           // Effacer la zone client et la repeindre
           InvalidateRect(HWindow,NULL,TRUE) ;
                UpdateWindow(HWindow) ;
        }//fin else is.bad()

}//fin focntion
//<< >> << >> << >>Methode de gestion des flux
//-------------
//        NOM DE LA METHODE:OpenFileInExistantEnSortie
//
//Parametres:
//Renvoie:
//Donnees de la classe utilisee: NomFichier, IsNewFile(M), Isdirty(M), SchemaEstOuvert(M)
//
//Commentaires:
//on force la fenetre à se redessiner apres la lecture du fichier
//

void TFenPrinc::OpenFileInExistantEnSortie()
{//début fonction

        ofpstream
                        os (NomFichier ); // ouvre un flux en sortie

        char  bufofp[180] ; strcpy(bufofp,"Impossible d'ouvrir le fichier ");
        strcat(bufofp,NomFichier) ; strcat(bufofp," en écriture !!");
        if( os.bad() )
                MessageBox(HWindow,bufofp,
```

```
                "O Etoile - Erreur Fichier",MB_OK|MB_ICONEXCLAMATION) ;
  else
        {//début else os.bad()

        // MessageBox(HWindow.NomFichier,
        //"Ce fichier existe MAINTENANT Ouvert en écriture",MB_OK|MB_ICONEXCLAMATION) ;
           DeleteAll() ;
           //afficher la boite liste de gauche
                ShowWindow(Liste->HWindow, SW_SHOWNORMAL) ;

           //afficher le nom du fichier en haut de la fenetre principale
           char buftitle[300]; strcpy(buftitle, "O Etoile - ");
           strcat(buftitle,NomFichier);
                SetCaption(buftitle) ;

                IsNewFile = IsDirty = FALSE ;

                GrapheStatique = new TGrapheStatique() ;
     DicoClasses   = new TDicoClasses () ;
                ConteneurLiensHERIT= new TConteneurLiensHERIT() ;
                ConteneurLiensCOMPOS = new TConteneurLiensCOMPOS();
                ConteneurLiensREFER = new TConteneurLiensREFER();
                ConteneurProprietes= new TTableauProprietesDsDom(50,0,10);
                ConteneurDomaines = new TTableauDomaines(50,0,10);


        SchemaEstOuvert = TRUE ;
           // Effacer la zone client et la repeindre
           InvalidateRect(HWindow,NULL,TRUE) ;
                UpdateWindow(HWindow) ;

        }// fin  else os.bad()

}//fin fonction
//-------------
//        NOM DE LA METHODE: TFenPrinc::SaveFile()
//
//Parametres:
//Renvoie:
//Donnees de la classe utilisee: NomFichier, IsNewFile(M), IsDirty(M)
//
//Commentaires:
//
//
void TFenPrinc::SaveFile()
{
ofpstream
os (NomFichier); // ouvre un flux en sortie
if( os.bad() )
        MessageBox(HWindow, "Impossible d'ouvrir le fichier en écriture!!!",
        "O Etoile - Erreur Fichier",MB_OK | MB_ICONEXCLAMATION) ;
else
        {
        write(os) ;
        os.close() ;
        IsNewFile = IsDirty = FALSE ;
        }
}
//-------------
//        NOM DE LA METHODE: TFenPrinc::SaveFileAs()
```

```
//
//Parametres:
//Renvoie:
//Donnees de la classe utilisee: NomFichier(M)
//
//Commentaires:
//
//
int TFenPrinc::SaveFileAs()
{

if( IsNewFile) strcpy(NomFichier,"") ;
if (GetApplication()->ExecDialog( new TFileDialog(this, SD_FILESAVE,
                    strcpy(NomFichier,"*.oet") ) ) == IDOK)
          {
          if( IsNewFile ) SplitCheminFichier(NomFichier);
          SaveFile() ;
          //afficher le nom du fichier en haut de la fenetre principale
          char bufsave[300]; strcpy(bufsave, "O Etoile - ");
          strcat(bufsave,NomFichier);
          SetCaption(bufsave) ;
          return IDOK ;
          }
return IDCANCEL ;
}


//[]-> Methodes d'information
//-------------
//          NOM DE LA METHODE: TFenPrinc::GetClassName ()
//
//Parametres:
//          -
//Renvoie: "TFenPrinc"
//
//Donnees de la classe utilisee:
//

LPSTR TFenPrinc::GetClassName ()
{
return "TFenPrinc" ;
}

//->[] Methodes de manipulation
//-------------
//          NOM DE LA METHODE: TFenPrinc::CanClose()
//
//Parametres:
//          -
//Renvoie: OK ou non pour fermer
//Donnees utilisees                                   (M) indique une modif de la donnee
//          -IsDirty
//Methodes appelees:
//
//Commentaires:
//
//
BOOL TFenPrinc::CanClose()
{//début fonction
```

```
if(!IsDirty)
        return TRUE ;// OK pour fermer
else
 {//début else
        char bufcanclose[180] ; strcpy(bufcanclose,file);
        strcat(bufcanclose,ext);
        strcat(bufcanclose," a été modifié\n"
                        "Sauvegarder la base avant de la fermer ?");
        int res = MessageBox(HWindow,bufcanclose,"O Etoile",
                                                MB_YESNOCANCEL|MB_ICONQUESTION);

        switch (res)
    {//début switch
                case IDCANCEL:  return FALSE;
                case IDYES:     if( IsNewFile)
                                                {
                                                if (SaveFileAs() !=IDOK)
                                                      return FALSE;
                                                }
                else
                                                SaveFile() ;
                break;

                case IDNO :    break;
          }//fin switch
        }//fin else
return TRUE ;
}//fin fonction

//--------------
//        NOM DE LA METHODE: TFenPrinc::GetWindowClass
//
//Parametres:
//        -WNDCLASS & UneFenetre
//Renvoie:
//Donnees utilisees                                (M) indique une modif de la donnee
//        -UneFenetre.hIcon
//Methodes appelees:
//
//Commentaires:
//
//
 void TFenPrinc::GetWindowClass (WNDCLASS & UneFenetre)
{
TWindow::GetWindowClass(UneFenetre) ;
UneFenetre.hIcon = LoadIcon (GetApplication()->hInstance, "ICON_OUTIL") ;
}

//--------------
//        NOM DE LA METHODE: TFenPrinc::SetupWindow()    (Surcharge)
//
//Parametres:
//        -
//Renvoie:
//Donnees utilisees                                (M) indique une modif de la donnee
//        -Liste
//Methodes appelees:Liste->AddString, Liste->SetSelIndex, SetFocus
//        -TWindow::SetupWindow
//Commentaires:
//initialisation des boites listes de la fenetre principal
```

```
//
void TFenPrinc::SetupWindow()
{
TWindow::SetupWindow() ;
Liste->AddString("Classe") :
Liste->AddString("Statique") ;
Liste->AddString("Dynamique") ;
Liste->SetSelIndex(0) ;
EnableKBHandler(); // autorise le passage du focus d' entrée
SetFocus(Liste->HWindow); // focus d'entée sur la boite de liste
bAide = FALSE ;
}
//-------------
//        NOM DE LA METHODE: TFenPrinc::KillAChild
//
//Parametres: néant
//        -
//Renvoie: rien
//Donnees utilisees                                    (M) indique une modif de la donnee
//        -Liste et RightBoiteListe de TFenPrinc
//Methodes appelees: ShutDownWindow
//Commentaires:
//Méthode permettant de tuer une fenêtre fille de TFenPrinc à la suite
//du métaévement  Base/Fermer Base
//

void    TFenPrinc::KillAChild(Pvoid child, Pvoid)
{  if (child ==Liste ) return ;
            if (child ==RightBoiteListe) return ;
             (( PTWindowsObject )child) -> ShutDownWindow() ;
            //effacer  le nom de l'ancienne base du titre de la fenetre principale
            SetCaption("Outil O Etoile") ;
}

//-------------
//        NOM DE LA METHODE:
//  TFenPrinc::MajFensConcerneParSupprClass
//
//Parametres:
//        -
//Renvoie: rien
//Donnees utilisees                                    (M) indique une modif de la donnee
//Methodes appelees:
//Commentaires:
//lors d'une suppression concernant une classe
//mettre à jour le contenu graphique des fenêtres qui
//mentionnent cette classe dans leur zone client ; elle possède
// comme argument TListeDeLiens*

void TFenPrinc::MajFensConcerneParSupprClass
                                                                    (TListeDeLiens* plistLiensExtr)
{//début fonction

 //on envoie un message WM_PAINT TFenGrapheStatique car cette fenêtre
 //contient toutes les icônes de classe
 InvalidateRect(
                                    GrapheStatique->FenGrapheStatique->HWindow,
                                    NULL,TRUE);
```

```
//envoyer un message WM_PAINT aux fenêtres dont la classe courante est
//une classe extrémité pour un lien donné (HERITAGE,COMPOSITION,
//REFERENCE ....
RContainerIterator listeIteratorRefreshAll =plistLiensExtr->
                                        initIterator();
while ( int(listeIteratorRefreshAll) != 0 )
 {//début while
 TLienConceptuel & rlienconRefreshAll =
          (TLienConceptuel &) listeIteratorRefreshAll++;
        if ( rlienconRefreshAll!= NOOBJECT )
        {//début if
        InvalidateRect(
                (rlienconRefreshAll.OrigineLien())->FenSchemaClasse->HWindow,
                                NULL,TRUE) ;
        }//fin if
 }//fin while
delete &listeIteratorRefreshAll;


}//fin fonction
//-------------
//        NOM DE LA METHODE:
// TFenPrinc::MajFensConcerneParModifClass
//
//Parametres:
//        -
//Renvoie: rien
//Donnees utilisees                                (M) indique une modif de la donnee
//Methodes appelees:
//Commentaires:
//lors d'une modification concernant une classe(par exemple renommage
//mettre à jour le contenu graphique des fenêtres qui
//mentionnent cette classe dans leur zone client ;

void TFenPrinc::MajFensConcerneParModifClass(
                                                        TSchemaClasse * pClasseRefresh)
{//début fonction

//on envoie un message WM_PAINT TFenGrapheStatique car cette fenêtre
//contient toutes les icônes de classe
InvalidateRect( GrapheStatique->FenGrapheStatique->HWindow,
                                NULL,TRUE);


//envoyer un message WM_PAINT à la FenSchemaClasse de la classe courante
InvalidateRect( pClasseRefresh->FenSchemaClasse->HWindow,
                                NULL,TRUE);

//envoyer un message WM_PAINT aux fenêtres dont la classe courante est
//une classe extrémité pour un lien donné (HERITAGE,COMPOSITION,
//REFERENCE ....
TListeDeLiens*  listeRefreshAll = ConteneurLiensHERIT->
        LiensDtClasseEstExtremite(pClasseRefresh->NomClasse,HWindow);
RContainerIterator listeIteratorRefreshAll =listeRefreshAll->
                                        initIterator();
while ( int(listeIteratorRefreshAll) != 0 )
 {//début while
 TLienConceptuel & rlienconRefreshAll =
          (TLienConceptuel &) listeIteratorRefreshAll++;
        if ( rlienconRefreshAll!= NOOBJECT )
```

```
        {//début if
        InvalidateRect(
                (rlienconRefreshAll.OrigineLien())->FenSchemaClasse->HWindow,
                        NULL,TRUE) ;
        }//fin if
  }//fin while
delete &listeIteratorRefreshAll;

}//fin fonction

//-------------
//        NOM DE LA METHODE:
// void  RafraichirUneFenFille(Pvoid , Pvoid ) ;
//
//Parametres: néant
//        -
//Renvoie: rien
//Donnees utilisees                                (M) indique une modif de la donnee
//        -Liste et RightBoiteListe de TFenPrinc
//Methodes appelees:
//Commentaires:
//
//
void  TFenPrinc::RafraichirUneFenFille(Pvoid child, Pvoid )
{//début fonction

if (child ==Liste ) return ;
if (child ==RightBoiteListe) return ;

InvalidateRect(   (( PTWindowsObject )child) -> HWindow,NULL,TRUE  );
UpdateWindow(     (( PTWindowsObject )child) -> HWindow     );

}//fin fonction


//{{{{{{{{{{{{{{{{ Methodes de reponses a un evenement
//-------------
//        NOM DE LA METHODE: Trt_Quitter
//
//Evenement: quitter l'application
//Donnees de la classe utilisee:
//        -
//Methodes appelees: CloseWindow
//Commentaires:
//
//
void TFenPrinc::Trt_Quitter(RTMessage Msg)
{
if (bAide)
        {
   bAide = FALSE ;
        WinHelp (HWindow, "aide hlp", HELP_CONTEXT,Msg.WParam * 10) ;
        }
else
        CloseWindow() ;

}
//{{{{{{{{{{{{{{{{ Methodes de reponses a un evenement
//-------------
//        NOM DE LA METHODE: Trt_FermerBase
```

```
//
//Evenement: quitter l'application
//Donnees de la classe utilisee:
//        -
//Methodes appelees: CloseWindow
//Commentaires:
//
//

void                    TFenPrinc::Trt_FermerBase(RTMessage )
{//début fonction
if (IsDirty)
        {//début if
          char buffermer[180] ; strcpy(buffermer,file);
          strcat(buffermer,ext);
          strcat(buffermer," a été modifié\n"
                        "Sauvegarder la base avant de la fermer ?");
          int res = MessageBox(HWindow,buffermer,"O Etoile",
                                                MB_YESNOCANCEL|MB_ICONQUESTION);

          switch (res)
     {//début switch
                case IDCANCEL:  SaveBeforeClose=IDCANCEL ;
                        return ;

                case IDYES:     SaveBeforeClose=IDYES;
                        if( IsNewFile)
                {
                                                if (SaveFileAs() !=IDOK)
                                                        return ;
                                              }
                else
                        SaveFile() ;
                break;

                case IDNO :     SaveBeforeClose = IDNO ;
                        break;
          }//fin switch
        }//fin if
RightBoiteListe->ClearList() ;
//rendre invisible la boite liste de gauche
ShowWindow(Liste->HWindow, SW_HIDE) ;
//rendre invisible la boite liste de droite
ShowWindow(RightBoiteListe->HWindow, SW_HIDE) ;

 //delete  Liste ;
 //delete  RightBoiteListe  ;

/*
*
*
*
*
 delete  Verification ;
 delete  GrapheStatique ;
 delete  ConteneurLiensHERIT;
 delete  ConteneurLiensCOMPOS ;
 delete  DicoClasses ;
*
*
```

```
*/


/* avant d'inclure ce code munir chaque classe liée à TGrapheStatique d'un
destructeur adéquat
if (GrapheStatique!=NULL) delete GrapheStatique ;
else
 { MessageBox(HWindow,
   "GrapheStatique est un pointeur null",
          "Echec de la tentative de libération mémoire (delete)",
          MB_OK|MB_ICONSTOP);
          exit(1);
 }
*/
DeleteAll() ;
/*
prévoir aussi d'effacer toutes fenetres filles de TFenPrinc hormis les
deux boites listes de départ Vues et Liste des classes
*/
KillAllChildren() ;

// Efface l' écran
InvalidateRect(HWindow,NULL,TRUE) ;
UpdateWindow(HWindow) ;


}//fin fonction


void   TFenPrinc::SplitCheminFichier(LPSTR splitPath)
{
flags = fnsplit(splitPath,drive,dir,file,ext) ;
}


//-------------
//          NOM DE LA METHODE: Trt_Nouveau
//
//Evenement: item nouveau
//Donnees de la classe utilisee:
//          -IsNewFile, SchemaEstOuvert
//Objets crees:
//          -GrapheStatique
//Methodes appelees:RightBoiteListe->ClearList, RightBoiteListe->AddString, DeleteAll()
//          -InvalidateRect
//Commentaires:
//Force la fonction paint
//
void TFenPrinc::Trt_Nouveau(RTMessage Msg)
{//début fonction

 if (bAide == TRUE)
          {
    bAide = FALSE ;
          WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
          }
 else
 { // début autre que demande d'aide
```

```
//avant de faire quoi que ce soit fermer la base précédente avant de
//s'installer; pour cela simuler clic sur Base/Fermer par envoi de message
SendMessage(HWindow, WM_COMMAND,53,MAKELONG(0,0) );

//afficher la boite liste de gauche
ShowWindow(Liste->HWindow, SW_SHOWNORMAL) ;
DeleteAll() ;

//IsDirty = FALSE; // déja fait dans DeleteAll()
IsNewFile = TRUE ;strcpy(file,"noname") ; strcpy(ext,".oet") ;

Verification = new TVerification() ;
GrapheStatique = new TGrapheStatique() ;
DicoClasses   = new TDicoClasses () ;
ConteneurLiensHERIT= new TConteneurLiensHERIT() ;
ConteneurLiensCOMPOS = new TConteneurLiensCOMPOS();
ConteneurLiensREFER = new TConteneurLiensREFER() ;
ConteneurProprietes= new TTableauProprietesDsDom(50,0,10).
ConteneurDomaines = new TTableauDomaines(50,0,10);


/*
Liste = new TListBox(this, ID_Liste,50,30,220,270) ;
Liste->Attr.Style &= ~LBS_SORT ; // liste non tiée
Liste->Attr.Style |= WS_CAPTION | WS_SIZEBOX | WS_CLIPCHILDREN ;
Liste->Attr.Style &= ~WS_MAXIMIZEBOX ; // lui oter sa case d'agrandissement
Liste->Attr.Style &= ~WS_VISIBLE ; // invisible au départ
Liste->Title = "Vues..." ;

RightBoiteListe = new TListBox(this,ID_RightBoiteListe,350,30,220,270) ;
RightBoiteListe->Attr.Style |= WS_CAPTION | WS_SIZEBOX |WS_CLIPCHILDREN ;
RightBoiteListe->Attr.Style &= ~WS_MAXIMIZEBOX ; // lui oter sa case d'agrandissement
RightBoiteListe->Attr.Style &= ~WS_VISIBLE; // invisible au départ
*/

SchemaEstOuvert = TRUE ;

//écrit noname dans la barre de titre
char bufnouv[60] ; strcpy(bufnouv,"O Etoile - noname");
strcat(bufnouv,".oet") ;
SetCaption(bufnouv) ;
// Efface l' écran
InvalidateRect(HWindow,NULL,TRUE) ,
UpdateWindow(HWindow) ;
} // fin de autre que demande d'aide

}//fin fonction
//----------
//        NOM DE LA METHODE:Trt_Ouvrir
//
//Evenement: item nouveau
//Donnees de la classe utilisee:
//        -IsNewFile, SchemaEstOuvert
//Objets crees:
//        -GrapheStatique
//Methodes appelees:RightBoiteListe->ClearList, RightBoiteListe->AddString, DeleteAll()
//        -InvalidateRect
//Commentaires:
//Force la fonction paint
//
```

```
void TFenPrinc::Trt_Ouvrir(RTMessage Msg)
{
if (bAide == TRUE)
        {
   bAide = FALSE ;
        WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
        }
else
 { // début autre que demande d'aide

  //avant de faire quoi que ce soit fermer la base précédente avant de
  //s'installer; pour cela simuler clic sur Base/Fermer par envoi de message
  SendMessage(HWindow, WM_COMMAND,53,MAKELONG(0,0) );

  //ici on regarde le résultat de l'envoi de message précédent(Trt_Fermer
  // au cas où le développeur a invoqué Base/Ourir pour ouvrir une
  //nouvelle base , la base courante étant modifiée,le logiciel lui
  //demande s'il veut sauver les modifications de la base courante
  //avant d'ouvrir une autre base

  switch (SaveBeforeClose) //attention initialiser correctement
          {//début switch      //SaveBeforeClose
                case IDCANCEL:  SaveBeforeClose=VALEURPARDEFAUT;
                                        return ;

                case IDYES:     SaveBeforeClose=VALEURPARDEFAUT;
                                        break ;

                case IDNO :     SaveBeforeClose =VALEURPARDEFAUT;
                case VALEURPARDEFAUT:
                                        int res ;
                                strcpy(NomFichier,"*.oet") ;
                                        res = GetApplication()->ExecDialog(
                                                new TFileDialog(this,SD_FILEOPEN,
                                   NomFichier)) ;
                                        if(res ==IDOK)
                                                {
                                                        SplitCheminFichier(NomFichier);

                                                        OpenFile() ;
                                                        //ShowWindow(Liste->HWindow,
SW_SHOWNORMAL) ;
                                                }
                                        else
                                                        SchemaEstOuvert = FALSE ;


          }//fin switch


  } // fin de autre que demande d'aide
}

//-------------
//        NOM DE LA METHODE: TFenPrinc::Trt_Enregistrer
//
//Evenement: item Enregistrer
//Donnees de la classe utilisee:
//        -
```

```
//Methodes appelees:
//Objets crees:
//        -
//Commentaires:
//
//
void TFenPrinc::Trt_Enregistrer(RTMessage Msg)
{
if (bAide == TRUE)
        {
   bAide = FALSE ;
        WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
        }
else
{ // début autre que demande d'aide
if( IsNewFile) //la base est sans nom
        SaveFileAs() ;
else
        SaveFile() ;
} // fin autre que demande d'aide
}

//-------------
//        NOM DE LA METHODE: Trt_EnregistrerSous
//
//Evenement: Item EnregistrerSous
//
//Donnees de la classe utilisee:
//        -
//Objet crees:
//        -
//Methodes appelees: saveFileAs
//Commentaires:
//
//
void TFenPrinc::Trt_EnregistrerSous(RTMessage Msg)
{
if (bAide == TRUE)
        {
   bAide = FALSE ;
        WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
        }
else
{ // début autre que demande d'aide
SaveFileAs() ;
//écrit le nom de la base dans la barre de titre
 char bufsous[300] ; strcpy(bufsous,"O Etoile - ");
 strcat(bufsous,NomFichier);
 SetCaption(bufsous) ;
} // fin autre que demande d'aide
}


//-------------
//        NOM DE LA METHODE:TFenPrinc::Trt_Cplusplus
//
//
//
//
//Objet crees:
```

```
//          -
//Methodes appelees:
//Commentaires:
//
//
//

void TFenPrinc::Trt_Cplusplus(RTMessage)
{//début fonction
void traiter_classes(HWND);
  traiter_classes(HWindow);
}//fin fonction
//-------------
//          NOM DE LA METHODE: Trt_Liste
//
//Evenement: evt emis par la 1ere listbox de la fen. princ.
//Donnees de la classe utilisee:
//          -RightBoiteListe
//          -bSchemaClasse(M),bGrapheDyn(M),bGrapheStatique (M)
//          -SchemaEstOuvert
//Objet crees:
//          -
//Methodes appelees:Liste->GetSelIndex, Liste->AddString, Liste->ClearString
//Commentaires:
// Remplissage des boite liste à l'aide des données de la base, pour Classe et Graphedynamique
// Pour GrapheStatique affichage du mot nomgraphestatique
//
void TFenPrinc::Trt_Liste(RTMessage Msg)
{
int Idx ;
/*if (bAide == TRUE)
          {
   bAide = FALSE ;
          WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
          }
else
{ // début autre que demande d'aide
*/
switch( Msg.LP.Hi )
          {
          case LBN_SELCHANGE :
          Idx = Liste -> GetSelIndex() ;
          switch (Idx )
          {
                    case 0: // sélection de l' item classes
                    if( SchemaEstOuvert)
                              {
                              bSchemaClasse = TRUE ;
                              // on cliqué sur classe
                              bGrapheStatique = bGrapheDynamique = FALSE ;
                              //rend visible la boite liste de droite et
                              //lui donne un titre correspondant au contexte
                              ShowWindow(RightBoiteListe->HWindow, SW_SHOWNORMAL) ;

                              RightBoiteListe->SetCaption("Liste des classes");
                              RightBoiteListe->ClearList() ;
                              RightBoiteListe->AddString("<créer>") ;
                              //Ecrire les noms des classes dans la boite liste de droite
                              DicoClasses->EcrireClassesDsBoiteListe(RightBoiteListe) ;
```

```
        }
                else
                        MessageBox(HWindow,"Aucune base n'est ouverte",
                                "Message",MB_OK|MB_ICONEXCLAMATION ) ;
                break ;

                case 1 : // sélection de l'item statique
                if( SchemaEstOuvert)
                        {
                        bGrapheStatique = TRUE ;
                        // on a cliqué sur statique
                        bSchemaClasse = bGrapheDynamique = FALSE ;
                        //rend visible la boite liste de droite et
                        //lui donne un titre correspondant au contexte
                        ShowWindow(RightBoiteListe->HWindow, SW_SHOWNORMAL) ;

                        RightBoiteListe->SetCaption("Le graphe statique");
                        RightBoiteListe->ClearList() ;
                        RightBoiteListe->AddString("NomGrapheStatique") ;
                        }
                else
                        MessageBox(HWindow,"Aucune base n'est ouverte",
                                "Message",MB_OK|MB_ICONEXCLAMATION ) ;

                break ;

                case 2: // sélection de l'item dynamique
                if( SchemaEstOuvert)
                        {
                        bGrapheDynamique = TRUE ;
                        bGrapheStatique = bSchemaClasse = FALSE ;
                        RightBoiteListe->ClearList() ;
                        //rend visible la boite liste de droite et
                        //lui donne un titre correspondant au contexte
                        ShowWindow(RightBoiteListe->HWindow, SW_SHOWNORMAL) ;
                        RightBoiteListe->SetCaption("Les Graphes Dynamiques");
                        RightBoiteListe->AddString("<créer>") ;
                        RightBoiteListe->AddString("GrapheDynamique 1") ;
                        RightBoiteListe->AddString("GrapheDynamique 2") ;
                        RightBoiteListe->AddString("GrapheDynamique 3") ;
                        RightBoiteListe->AddString("GrapheDynamique 4") ;
                        RightBoiteListe->AddString("GrapheDynamique 5") ;
                        RightBoiteListe->AddString("GrapheDynamique 6") ;
                        RightBoiteListe->AddString("GrapheDynamique 7") ;
                        RightBoiteListe->AddString("GrapheDynamique 8") ;
                        RightBoiteListe->AddString("GrapheDynamique 9") ;
                        }
                else
                        MessageBox(HWindow,"Aucune base n'est ouverte",
                                "Message",MB_OK|MB_ICONEXCLAMATION ) ;

                break ;
                }
        break ;
        }
//}// fin autre que demande d'aide
}

//-------------
```

```
//        NOM DE LA METHODE: Trt_SchemaStaticDynamic
//
//Evenement: ,mis par la 2eme listbox de la fen. princ.
//Donnees de la classe utilisee:
//        -
//Objet crees:
//        -TCreationClasseDial
//        -TSchemaClasse
//Methodes appelees: RightBoiteListe->GetSelIndex() , RightBoiteListe->AddString, GetSelString
//        -GrapheStatique->Appartient, GrapheStatique->AjouterClasse
//        -ViewFenetreClasse()
//        -Verification->UniciteClasse
//        -TrackPopupMenu(hMenuClasse)
//        -TrackPopupMenu(hMenuGraphDyn
//        -GrapheStatique->VisualiserFen
//Commentaires:
// gère la creation d'une classe dans les differents modeles
//
void TFenPrinc::Trt_SchemaStaticDynamic(RTMessage Msg)
{
int             Idx ;
switch( Msg.LP.Hi )
        {
        case LBN_SELCHANGE:
        Idx = RightBoiteListe->GetSelIndex() ;
        // cas du schéma de classe
        if( bSchemaClasse&& !bGrapheStatique && !bGrapheDynamique)
                {
                IndexClasse = Idx ;
                RightBoiteListe->GetSelString( NomClasse , sizeof NomClasse) ;
                switch(Idx)
                        {
                        case 0: // clic sur créer
                        int res ;
                        res = GetApplication()->ExecDialog( new TCreationClasseDial(this,
"CreationClasse")) ;
                        if( res ==IDOK)
                                {
                                if ( strlen(szNomClasseCree) )
                                        {
                                        // on pourra améliorer en ajoutant
                                        // d'autres controles
                                        // si le nom du schéma n'existe pas déjà dans le DicoClasses
                                        // l'inserer dans le DicoClasses
                                        // Visualiser sa fenetre SCHEMA CLASSE
                                        TSchemaClasse * p ;

                                        if(!(DicoClasses->EstMembre(szNomClasseCree,p)))
                                                {

                                                RightBoiteListe->AddString(szNomClasseCree) ;
                                                p->ViewFenetreClasse() ;
                                                IsDirty = TRUE ;
                                                DicoClasses->AjouterNouvelleEntree(p);
                                                //mise à jour de FenGrapheStatique pour qu'elle
                                                //se réaffiche en tenant compte de la nouvelle
                                                //classe créee.
        InvalidateRect(GrapheStatique->FenGrapheStatique->HWindow,  NULL,TRUE) ;
                                                UpdateWindow(
                                                GrapheStatique->FenGrapheStatique->HWindow) ;
```

```
                                        }
                                        //si le schéma de classe existe déjà le signaler
                                        //En effet tout nom de classe est unique à l' intérieur
                                // d' un schéma conceptuel (Controle de conformité par rapport au
                                        // modèle O Etoile
                                        else
                                                {
                                        Verification->UniciteClasse(HWindow,szNomClasseCree) ;
                                                }
                                        }
                                }
                        break ;
                        default: // clic sur une classe
                        lstrcpy(NomClassePRED, NomClasse) ;
                        POINT posCurseur;
                        GetCursorPos(&posCurseur);
TrackPopupMenu(hMenuClasse,TPM_CENTERALIGN,posCurseur.x,posCurseur.y,0,HWindow,NULL
) ;
                        break ;
                }
        }
        // cas du graphe statique
        if( !bSchemaClasse&& bGrapheStatique && !bGrapheDynamique)
                {
                switch( Idx )
                        {
                        case 0: // clic sur nom du graphe statique
                                GrapheStatique->VisualiserFen() ;
                                break ;
                        }
                }
        // cas du Graphe Dynamique
        if( !bSchemaClasse&& !bGrapheStatique && bGrapheDynamique)
                {
                IndexGraphDyn = Idx ;
                RightBoiteListe->GetSelString( NomGraphDyn , sizeof NomGraphDyn) ;
                switch( Idx )
                        {
                        case 0: // clic sur créer
                                MessageBox(HWindow,"<créer>","CLIC SUR ". MB_OK) ;
                                break ;
                        default: // clic sur un nom de graphe dynamique
                                TrackPopupMenu(hMenuGraphDyn,0,450,125,0,HWindow,NULL)
                                break ;
                        }
                }
        break ;
        }
}


//-------------
//        NOM DE LA METHODE  RenommerClasse
//
//Evenement: item RenommerClasse du menu flottant hClasse
//Donnees utilisees:
//        -NomClasse, NomClassePRED
//        -bSchemaClasse, bGrapheStatique, bGrapheDynamique
//        -GrapheStatique->GrapheHeritage (M)
//Objet crees:
```

```
//        -TRenClasseDial
//        -
//Methodes appelees:
//        -GrapheStatique->Appartient
//        -RightBoiteListe->DeleteString(IndexClasse), RightBoiteListe->AddString
//        -TSchemaClasse->FenSchemaClasse->SetCaption
//Commentaires:
//
//
void TFenPrinc:: RenommerClasse(RTMessage )
{//début fonction

char buf[80] ;
/*
if (bAide == TRUE)
        {
   bAide = FALSE ;
        WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
        }
else
{ // début autre que demande d'aide
*/
if( bSchemaClasse&& !bGrapheStatique && !bGrapheDynamique)
        {//début if
        int res ;
        res = GetApplication()->ExecDialog( new TRenClasseDial(this, "RenClasse")) ;
        if( res==IDOK)
                {//début if
                if ( strlen(NomClasse) )
                        {//début if
                        TSchemaClasse * pschema ;
                        if(DicoClasses->EstMembre(NomClasse,pschema))
                                {//début if
                                lstrcpy(buf,NomClasse);lstrcat(buf," existe déjà dans le dictionnaire ")
;
                                for(int i = 0; i<= 3;i++) MessageBeep(0) ;
                                MessageBox(HWindow,buf,"Renommage impossible !!!",
                                        MB_ICONEXCLAMATION|MB_OK) ;
                                }//fin if
                  else
                        {//début else
                        RightBoiteListe->DeleteString(IndexClasse) ;
                        RightBoiteListe->AddString(NomClasse) ;
                        IsDirty = TRUE ;
                        DicoClasses->RenommerClasse(NomClassePRED,NomClasse ) ,

                        MajFensConcerneParModifClass(pschema) ;

          }//fin else
                    }//fin if
           }//fin if
   }//fin if
//} // fin autre que demande d'aide
}//fin fonction

//-------------
//        NOM DE LA METHODE: EditerSchemaClasse
//
//Evenement: item editer du menu flottant hClasse
//Donnees de la classe utilisee:
```

```
//          -bSchemaClasse, bGrapheStatique, bGrapheDynamique
//Objet crees:
//          -TSchemaClasse(en local)
//Methodes appelees:
//          -GrapheStatique->Appartient
//          -GrapheStatique->GrapheHeritage[ind]).Classe ->ViewFenetreClasse
//Commentaires:
//
//
void TFenPrinc::EditerSchemaClasse(RTMessage )
{
/*
if (bAide == TRUE)
          {
   bAide = FALSE ;
          WinHelp (HWindow, "aide hlp", HELP_CONTEXT, Msg.WParam * 10) ;
          }
else
{ // début autre que demande d'aide
*/
if( bSchemaClasse&& !bGrapheStatique && !bGrapheDynamique)
 {
          TSchemaClasse          * p ;
          //p = new TSchemaClasse(NomClasse) ;
          // si le nom du schéma existe dans le GRAPHE STATIQUE
          // Visualiser sa fenetre SCHEMA CLASSE
          if((DicoClasses->EstMembre(NomClasse,p)))
            p->ViewFenetreClasse() ;

 }// fin autre que demande d'aide
}//fin fonction
//-------------
//          NOM DE LA METHODE: TFenPrinc::DeleteStrngInRightLstBox(
//
//Evenement: item editer du menu flottant hClasse
//Donnees de la classe utilisee:
//
//Objet crees:
//
//Methodes appelees:
//
//
//Commentaires:
//
//

void TFenPrinc::DeleteStrngInRightLstBox( int Idx )
{//début fonction

RightBoiteListe->DeleteString(Idx) ;


}//fin fonction

//-------------
//          NOM DE LA METHODE  TFenPrinc::SupprimerClasse
//
//Evenement: item supprimer du menu flottant hclasse
//Donnees de la classe utilisee:
//          -bSchemaClasse bGrapheStatique bGrapheDynamique
```

```
//          -IsDirty(M)
//Objet crees:
//          -TDropClasseDial
//Methodes appelees:
//          -RightBoiteListe->DeleteString
//Commentaires:
//dialogue de suppression d'une classe
//
void TFenPrinc::SupprimerClasse(RTMessage )
{
/*
if (bAide == TRUE)
          {
   bAide = FALSE ;
          WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
          }
else
{ // début autre que demande d'aide
*/
if( bSchemaClasse&& !bGrapheStatique && !bGrapheDynamique)
          {//début if ( bSchemaClasse&& !bGrapheStatique && !bGrapheDynamique)
          int res ;
          res = GetApplication()->ExecDialog( new TDropClasseDial(this, "DropClasse")) ;
          if( res==IDOK)
                    {//début if( res==IDOK)
                     //je récupère cette liste de classe pour leur envoyer un message
       //WM_PAINT après la mort de la classe pour leur zone client
                     TListeDeLiens*  listeRefreshAll = ConteneurLiensHERIT->
                          LiensDtClasseEstExtremite(NomClasse,HWindow);

                     DicoClasses->DetruireClasseEtDependances(NomClasse,IndexClasse) ;
                     IsDirty = TRUE ;
                     MajFensConcerneParSupprClass(listeRefreshAll );
                     //RafraichirToutesLesFenFilles();
                    }//fin  if( res==IDOK)
          }//fin if( bSchemaClasse&& !bGrapheStatique && !bGrapheDynamique)
//}// fin autre que demande d'aide
}

//-------------
//          NOM DE LA METHODE: RenommerGrapheDynamique
//
//Evenement: item renommer du menu flottant hdynamique
//Donnees de la classe utilisee:
//          -bSchemaClasse bGrapheStatique bGrapheDynamique
//Objet crees:
//          -TRenGraphDynDial
//Methodes appelees:
//          -RightBoiteListe->DeleteString, RightBoiteListe->AddString
//          -
//Commentaires:
// incomplete .....
//
void TFenPrinc.: RenommerGrapheDynamique(RTMessage )
{
/*
if (bAide == TRUE)
          {
   bAide = FALSE ;
          WinHelp (HWindow, "aide.hlp", HELP_CONTEXT,Msg.WParam * 10) ;
```

```
        }
else
{ // début autre que demande d'aide
*/
if( !bSchemaClasse&& !bGrapheStatique && bGrapheDynamique)
        {
        int res ;
        res = GetApplication()->ExecDialog( new TRenGraphDynDial(this, "RenGraphDyn")) ;
        if( res==IDOK)
                {
                if ( strlen(NomGraphDyn) )
                        {
                        RightBoiteListe->DeleteString(IndexGraphDyn) ;
                        RightBoiteListe->AddString(NomGraphDyn) ;
                        }
                }
        }
//}// fin autre que demande d'aide
}

//-------------
//        NOM DE LA METHODE. SupprimerGrapheDynamique
//
//Evenement: item supprimer du menu hdynamique
//Donnees de la classe utilisee:
//        -bSchemaClasse bGrapheStatique  bGrapheDynamique
//        -IndexGraphDyn
//Objet crees:
//        -TDropGraphDynDial
//Methodes appelees:
//        -RightBoiteListe->DeleteString
//Commentaires:
//incomplete .....
//
void TFenPrinc:: SupprimerGrapheDynamique(RTMessage )
{
if( !bSchemaClasse&& !bGrapheStatique && bGrapheDynamique)
        {
        int res ;
        res = GetApplication()->ExecDialog( new TDropGraphDynDial(this, "DropGraphDyn")) ;
        if( res==IDOK)
                {
                RightBoiteListe->DeleteString(IndexGraphDyn) ;
                }
        }
}

//-------------
//        NOM DE LA METHODE. CMAbout
//
//Evenement: item about du menu principal
//Donnees de la classe utilisee:
//        -
//Objet crees:
//        -TAboutDialog
//Methodes appelees:
//        -
//Commentaires:
//
//
```

```
void TFenPrinc::CMAbout(RTMessage Msg)
{
if (bAide == TRUE)
        {
   bAide = FALSE ;
        WinHelp (HWindow, "aide.hlp", HELP_CONTEXT, Msg.WParam * 10) ;
        }
else
{ // début autre que demande d'aide

GetApplication()->ExecDialog(new TAboutDialog(this,"About")) ;
}// fin autre que demande d'aide
}


//-------------
//        NOM DE LA METHODE: CMIndex
//
//Evenement: item menu princ index de l'aide
//Donnees de la classe utilisee:
//        -
//Methodes appelees:
//        -WinHelp
//
void TFenPrinc::CMIndex (RTMessage )
{
WinHelp (HWindow, "aide.hlp", HELP_INDEX, 0L) ;
}


//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE: TVerification
//*************************************************************************
//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
//->[] Methodes de manipulation
//-------------
//        NOM DE LA METHODE: UniciteClasse
//
//Parametres:
//        -HWND & hwind: handle de fenetre
//        -TSchemaClasse * psc, .pointeur sur TSchemaClasse existant
//        -WORD indice: indice de la classe dans la structure de données
//Renvoie:
//Donnees utilisees                                (M) indique une modif de la donnee
//        -
//Methodes appelees:
//        -MessageBox
//Commentaires:
//Cette méthode ne gère pas le controle, elle affiche seulement un message
//a l'utilisateur
//
void TVerification::UniciteClasse(HWND & hwind,LPSTR sznomCLASSE)
{
char buf[80] ;
lstrcpy(buf,sznomCLASSE) ;
lstrcat(buf," existe déjà ") ;
for(int i = 0; i<=3;i++) MessageBeep(0) ;
MessageBox(hwind,buf,"Controle de conformité: Violation unicité nom de
classe",MB_ICONSTOP|MB_OK) ;
```

```
}


//*************************************************************************
//                METHODES DE LA CLASSE: TApp
//*************************************************************************

//->[] Methodes de manipulation
//-------------
//          NOM DE LA METHODE: InitMainWindow     (surcharge)
//
//Parametres:
//        -
//Renvoie:
//Donnees utilisees                                  (M) indique une modif de la donnee
//        -
//Objets Crees:
//      - TFenPrinc
//Methodes appelees:
//        -
//Commentaires:
//
//
void TApp::InitMainWindow()
{
MainWindow = new TFenPrinc(Name) ;
}

//~~~~~~~~~~~~~~~~~ Surcharge d'opérateurs~~~~~~~~~~~~~~~~~~
//les operateurs de flux sont surchargés pour les objets streamable

inline Ripstream operator>>( Ripstream is,RTFenPrinc tw)
{
return is >> (RTStreamable) tw ;
}

inline Ripstream operator>>( Ripstream is,RPTFenPrinc tw)
{
return is >> (RPvoid) tw ;
}

inline Ropstream operator<< ( Ropstream os,RTFenPrinc tw)
{
return os << (RTStreamable) tw ;
}

inline Ropstream operator<< ( Ropstream os,PTFenPrinc tw)
{
return os << (PTStreamable) tw ;
}

// le code suivant doit être exécuté une fois et une fois seulement
// pour toute classe flux que l'on définit. ce code est normalement placé
// juste avant la boucle principale de WinMain()

// Enregistre la classe flux TFenPrinc

TStreamableClass RegTFenPrinc("TFenPrinc",TFenPrinc::build, __DELTA(TFenPrinc)) ;

TStreamableClass RegTIconeClasse("TIconeClasse",TIconeClasse::build, __DELTA(TIconeClasse)) ;
```

```
TStreamableClass RegTSchemaClasse("TSchemaClasse",TSchemaClasse::build,
__DELTA(TSchemaClasse)) ;

TStreamableClass RegTGrapheStatique("TGrapheStatique",TGrapheStatique::build,
__DELTA(TGrapheStatique)) ;

TStreamableClass RegTFenGrapheStatique("TFenGrapheStatique",TFenGrapheStatique::build,
__DELTA(TFenGrapheStatique)) ;

TStreamableClass RegTDicoClasses("TDicoClasses",TDicoClasses::build, __DELTA(TDicoClasses))
;

TStreamableClass
RegTNomEtObjetclasseAssoc("TNomEtObjetclasseAssoc".TNomEtObjetclasseAssoc::build,
__DELTA(TNomEtObjetclasseAssoc)) ;

TStreamableClass RegTCommentaire("TCommentaire",TCommentaire::build,
__DELTA(TCommentaire)) ;

TStreamableClass RegTFenCommentaire("TFenCommentaire", TFenCommentaire::build,
                                    __DELTA(TFenCommentaire));



//=*=*=*=*=*=*=*=*=*=*PROGRAMME PRINCIPAL=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=

int PASCAL WinMain(HANDLE hInst, HANDLE hPrevInst, LPSTR lpCmdLine, int nCmdShow)
        {
        TApp App ("Outil O Etoile", hInst, hPrevInst, lpCmdLine, nCmdShow) ;
        App.Run() ;
        return App.Status ;
        }




//============================================================================
//              NOM DE FICHIER: CON_TDA.HPP
//
//
//Liste des classes:
//              .Classes logicielles
//                      -TDicoClasses
//                      -TDicoEntree
//                              -TConteneurLiens
//                              -TConteneurLiensHERIT
//                              -TConteneurLiensCOMPOS
//                              -TConteneurLiensREFER
//                      -TNomEtObjetclasseAssoc
//                      -TNomEtListeDoubleAssoc
//                              -TAttribEtObjetAttribAssoc
//                      -TDeuxListsOrigExtr
//                              -TArrayWithHolesOfLinks
//                              -TTableauProprietesDsDom
//                              -TTableauDomaines
//                              -TEnsembleConcept
//              .Classes interfaces
```

```
//                                    -
//                                    -
//=============================================================================

//*********REMARQUES.

//*********Indications compilateur
#ifndef WIN30
#define WIN30
#endif

#ifndef _CON_TDA
#define _CON_TDA
//*********Liste des fichiers inclus
#include "objets.hpp"
#include <dict.h>
#include <assoc.h>
#include <array.h>
#include <set.h>
#include <OBJSTRM.H>
#include <windows.h>

#include "liens.hpp"

//*********Declaration des classes externes utilisees

//*****************************************************************************
//*****************************************************************************
//                    NOM DE LA CLASSE: TDicoClasses
//
//Description:
//TDicoClasses est une classe conteneur contenant un ensemble
//d' associations instances de la classe TNomEtObjetclasseAssoc
// chaque objet TNomEtObjetclasseAssoc est un couple nom de classe
// (clé de l' association) et un objet TSchemaClasse (valeur de l'
//association
//
//Parametres de creation:
//unsigned sz  taille de départ du dictionnaire
//
//Comportement du destructeur:
//
//Liste des ancetres:Dictionary,Set,Bag,Collection,Container,Object
//,TShouldDelete, TStreamable
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-----------------------------------------------------------------------

class TSchemaClasse ;
_CLASSDEF(TDicoClasses)
class    TDicoClasses: public TStreamable ,public Dictionary

{
 //                                  ----------PRIVE----------

//->->->->->->->->->->-> Liste des methodes
```

```
//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;


        TDicoClasses(StreamableInit) ;

//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TDicoClasses( unsigned sz = DEFAULT_HASH_TABLE_SIZE ) :        Dictionary(sz)
    {
        }

//[]->Methodes d'information
 BOOL EstMembre(LPSTR,TSchemaClasse  * &);
 BOOL EstMembre(LPSTR,TSchemaClasse* &,BOOL&,HWND ) ;
 TSchemaClasse  * IdentifierIconeClasse(RTMessage,RECT &);
//->[]Methodes de manipulation
void EcrireClassesDsBoiteListe(TListBox          *);
void AjouterNouvelleEntree(TSchemaClasse  * )  ;
void RenommerClasse(LPSTR, LPSTR);
void DetacherClasse(LPSTR )  ;
void DetruireClasseEtDependances(LPSTR,int);

//void ReconstitueLiensHeritage() ;
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
        static PTStreamable build() ;
        virtual void write(Ropstream ) ;
        virtual Pvoid read(Ripstream ) ;

};


//****************************************************************************
//****************************************************************************
//                NOM DE LA CLASSE: TNomEtObjetclasseAssoc
//
//Description:
//
//Parametres de creation:
//Object & cleNomClasse :   clé de l' association objet de type String
//Object & valeurObjetClasse : valeur de l'association un objet de
//TSchemaClasse
//Comportement du destructeur:
//
//Liste des ancetres: Association, TStreamable, Object, TShouldDelete
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
```

```
//---------------------------------------------------------------------



_CLASSDEF(TNomEtObjetclasseAssoc)
class    TNomEtObjetclasseAssoc: public Association, public TStreamable
{


//                                  ----------PRIVE----------

//->->->->->->->->->->-> Liste des methodes
//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;

        TNomEtObjetclasseAssoc(StreamableInit) ;

//                                  ++++++++++++++++++PUBLIQUE++++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TNomEtObjetclasseAssoc(Object & cleNomClasse,Object & valeurObjetClasse )
 : Association(cleNomClasse, valeurObjetClasse )
                {
                }


//[]->Methodes d'information

//->[]Methodes de manipulation
void RenommerClasse(LPSTR);

//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
        static PTStreamable build() ;
        virtual Pvoid read(Ripstream ) ;
        virtual void write(Ropstream ) ;

};

//**************************************************************************
//**************************************************************************
//                NOM DE LA CLASSE: TConteneurLiens
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
```

```
//Classes amies:
//
//Operateurs surcharges:
//---------------------------------------------------------------------------


_CLASSDEF(TConteneurLiens)
class  TDicoEntree ;
class  TArrayWithHolesOfLinks ;
class  TNomEtListeDoubleAssoc ;
class  TLienConceptuel ;
class TLienHeritageConceptuel ;
class    TConteneurLiens : public TStreamable
{
//                                  ----------PRIVE----------
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes

//->[]Methodes de manipulation
TListeDeLiens* Concat_ListeOrig_ListeExtr(TListeDeLiens*,TListeDeLiens*);

//<< >> << >> Methode de gestion des flux
virtual const Pchar streamableName() const ;
                    TConteneurLiens(StreamableInit) ;


//                                  +-+-+-+-+-+-+-+-+-+PROTEGE+-+-+-+-+-+-+-+-+
protected:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees
TDicoEntree  * pdicoEntree ;
TArrayWithHolesOfLinks* ptableauliens ;


//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees
//on maintient une liste de liens d'héritage créés  à partir d'une
//TFenSchemaClasse pour pouvoir leur donner des points origine et
//extrémités de leur lien graphique associé dans TFenGrapheStatique
TListeDeLiens*  pLiensHeritDeTFenSchemaClasse ;
//on demande au conteneur de liens de maintenir une liste de classe
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage (cela permet des liens qui pendent)
TListeDeClasses* pClassesJustBeenRenamed ;
//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TConteneurLiens() ;

//[]->Méthodes d'information

//largeur totale superclasses de la classe en argument, sert dans
//TSchemaClasse::VisualiserSuperClasses pour la visualisation dans une
//TFenSchemaClasse des superclasses en progressant HORIZONTALEMENT
//à chaque nouvelle classe à afficher
int LargeurTotaleSuperClasses(HDC, LPSTR, HWND);
```

```
TListeDeLiens * LiensDtClasseEstOrigine(LPSTR szlienNomClasse,HWND ).
TListeDeLiens * LiensDtClasseEstExtremite(LPSTR NomDeClasse ,HWND);
TListeDeLiens * LiensDeLaClasse(LPSTR NomDeClasse, HWND);
void SuperClassesDe (TSetClasses  *,TSchemaClasse *, HWND) ;
//BOOL   LienEstMembre(TSchemaClasse * , TSchemaClasse *,HWND) ;
TLienConceptuel        *        IdentifierLien(RTMessage) ;

//->[]Methodes de manipulation
//void SupprimerLien(LPSTR,LPSTR,HWND) {}
void SupprimerLienConceptuel(TLienConceptuel *,HWND) ,
//void DetruireClasseEtDependances(LPSTR,HWND) ;

virtual void AddLinkSansToucherDicoEntree(LPSTR,LPSTR
                                                    ,HWND,Ripstream,int)
{}//implémentation par défaut
TLienHeritageConceptuel * AjouterLienHeritage(LPSTR,LPSTR,HWND);//méthode surchargée I
void AjouterLienHeritage(LPSTR,LPSTR,POINT,POINT,
            TPosLienGrSurIcone.
                                    TPosLienGrSurIcone,HWND);//méthode surchargée II
void AjouterLienConceptuel(TLienConceptuel * ,POINT,POINT,
                        TPosLienGrSurIcone,
                                    TPosLienGrSurIcone,HWND);

void DisplayAllLinks(HDC) ;

//donner une représentation graphique par défaut des liens crées à partir
//des TFenChemaClasse
void SetDefaultsGrLinksDeTFenSch(HDC) ;
void SetDefautsGrForOneHLink(TLienConceptuel *,HDC) ;

void AdjustGrLinksOnClassesRenamed(HDC) ;
void AdjustOneGrLinkOnRenamedClass(TSchemaClasse *, HDC);

void AjouterLienHdeTFenSCh(TLienConceptuel *) ;

//on demande au conteneur de liens de maintenir une liste de classe
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage (cela permet des liens qui pendent)
void AddClassJustRenamedInList(TSchemaClasse * ) ;

int MettreLienDsTableauLiens(TLienConceptuel *,HWND) ;
void AjouterAssocDsDicoEntree(LPSTR);
void RenommerClasse(LPSTR,LPSTR );
void MajListesOriginesEtExtremites(LPSTR,LPSTR,int,HWND) ;
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
static PTStreamable build() ;
virtual void write(Ropstream ) ;
virtual Pvoid read(Ripstream ) ;


};


//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TDicoEntree
```

```
//
//Description:
//TDicoEntree est une classe conteneur contenant un ensemble
//d' associations instances de la classe TNomEtListeDoubleAssoc
// chaque objet TNomEtListeDoubleAssoc est un couple nom de classe
// (clé de l' association) et un objet TSchemaClasse (valeur de l'
//association
//
//Parametres de creation:
//unsigned sz  taille de départ du dictionnaire
//
//Comportement du destructeur:
//
//Liste des ancetres:Dictionary,Set,Bag,Collection,Container,Object
//,TShouldDelete, TStreamable
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//------------------------------------------------------------------------

_CLASSDEF(TDicoEntree)
class  TDeuxListsOrigExtr ;
class    TDicoEntree: public Tstreamable, public Dictionary

{

//                                  ----------PRIVE----------

//->->->->->->->->->->-> Liste des methodes

//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;

        TDicoEntree(StreamableInit) ;

//                                  ++++++++++++++++++PUBLIQUE++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TDicoEntree( unsigned sz = DEFAULT_HASH_TABLE_SIZE ) :          Dictionary(sz)
     { }

//[]->Methodes d'information
TListeIndices  * ObtenirListeIndicesOrigines(LPSTR ,HWND);
TListeIndices  * ObtenirListeIndicesExtremites(LPSTR ,HWND);
//->[]Methodes de manipulation
void RenommerClasse(LPSTR, LPSTR );
void DetacherClasse(LPSTR,TDeuxListsOrigExtr * & );
void MajListesOriginesEtExtremites(LPSTR szOrigine,LPSTR szExtremite,
                                                int nIndArray,
                                                  HWND hwndmaj),
//OK<----Methodes d'autorisation
```

```
//<< >> << >> Methode de gestion des flux
        static PTStreamable build() ;
        virtual void write(Ropstream ) ;
        virtual Pvoid read(Ripstream ) ;

};
```

```
//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TNomEtListeDoubleAssoc
//
//Description:
//
//Parametres de creation:
//Object & cleNomClasse :   clé de l' association objet de type String
//Object & valeurObjetClasse : valeur de l'association un objet de
//TDeuxListsOrigExtr
//Comportement du destructeur:
//
//Liste des ancetres: Association, TStreamable, Object, TShouldDelete
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------

_CLASSDEF(TNomEtListeDoubleAssoc)
class    TNomEtListeDoubleAssoc: public Association, public TStreamable

{

//                                  ------------------PRIVE----------------

//->->->->->->->->->->->-> Liste des methodes

//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;

        TNomEtListeDoubleAssoc(StreamableInit) ;

//                                  ++++++++++++++++++++PUBLIQUE+++++++++++++++
public:
//->->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TNomEtListeDoubleAssoc(Object & cleNomClasse,Object & valeurDbleListe )
 : Association(cleNomClasse, valeurDbleListe)
                {
                }

//<< >> << >> Methode de gestion des flux
        static PTStreamable build() ;
```

```
        virtual Pvoid read(Ripstream ) ;
        virtual void write(Ropstream ) ;


};

//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TDeuxListsOrigExtr
//
//Description:
//
//Parametres de creation:
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------

_CLASSDEF(TDeuxListsOrigExtr)
class    TDeuxListsOrigExtr: public Object,public TStreamable

{
 friend TDicoEntree ;
 //                                  ------------------PRIVE----------------
 //@ @ @ @ @ @ @ @ @ @  Liste des donnees
 TListeIndices  * plisteOrigines ;
 TListeIndices  * plisteExtremites;

//->->->->->->->->->->-> Liste des methodes

//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;

        TDeuxListsOrigExtr(StreamableInit) ;

//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
public:

//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TDeuxListsOrigExtr()
{
 plisteOrigines = new TListeIndices();
 plisteExtremites = new TListeIndices();
}
//Méthodes virtuelles pures héritées de Object ( classes conteneurs)
//on les redéfinie ici pour avoir une classe concrète
virtual classType isA() const
{
 return originesExtremites;
}
virtual char _FAR *nameOf() const
{
```

```
 return "originesExtremites" ;
}
virtual hashValueType hashValue()  const
{
 hashValueType  valeurhachage = plisteOrigines-> hashValue() ;
 return valeurhachage ;
}
virtual int isEqual( const Object _FAR & testdblelist) const .
{
int resOrigines=
 plisteOrigines->
  isEqual(
        *(
 ((TDeuxListsOrigExtr & )testdblelist).plisteOrigines
        )
             )       :

int resExtremites=
 plisteExtremites->
  isEqual(
        *(
 ((TDeuxListsOrigExtr &)testdblelist).plisteExtremites
                       )
                     );
 return resOrigines && resExtremites ;
}

virtual void printOn( ostream & _FAR outputStream ) const
{

}

//[]->Methodes d'information
TListeIndices   * listeOrigines() const
{
return plisteOrigines ;
}
TListeIndices   * listeExtremites() const
{
return plisteExtremites;
}
//->[]Methodes de manipulation
void MajListeOrigine(LPSTR szOrig,int nIdArrayor,HWND hwdmajor ).
void MajListeExtremite(LPSTR szExtr,int nIdArrayex,HWND hwdmajex ).
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
        static PTStreamable build() ;
        virtual Pvoid read(Ripstream ) ;
        virtual void write(Ropstream ) ;

};

class  TArrayWithHolesOfLinks : public Array , public TStreamable
{

 //                              ------------------PRIVE----------------
 //@ @ @ @ @ @ @ @ @ @ Liste des donnees
TConteneurLiens                 *       ContenurLiensDeArrayHoles ;
//->->->->->->->->->->-> Liste des methodes
```

```
//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;

        TArrayWithHolesOfLinks(StreamableInit) ;

//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees

//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TArrayWithHolesOfLinks(TConteneurLiens * pContDeArray, int upper, int lower = 0, sizeType aDelta
= 0 )
                    : Array( upper, lower, aDelta )
 {
 ContenurLiensDeArrayHoles= pContDeArray;
 }
 TArrayWithHolesOfLinks(int upper, int lower = 0, sizeType aDelta = 0 )
          :Array( upper, lower, aDelta )
         { }
//vvvvvvvvvvvvvvvvvv  Méthodes virtuelles pures héritée de Object
//                possédant uneimplémentation ici car
//                TLienConceptuel est      aussi une classe concrète
 virtual classType isA() const
 {
 return tarrayWithHolesOfLinks;
 }

 virtual char _FAR *nameOf() const
 {
 return "TArrayWithHolesOfLinks";
 }


//[]->Methodes d'information
unsigned int lastElementInd ( )
{
  return lastElementIndex ;
}
//retourne l'indice dans le tableau correspondant à l'objet tof
int TrouverIndiceDsTableau(const Object & tof)
{
  return find(tof) ;
}
//->[]Methodes de manipulation
 virtual void  Ajouter ( Object &, int & ) ;
 int AjouterDomaine(LPSTR , LPSTR ) ;
 int AjouterPropriete(LPSTR,LPSTR);
 virtual void detach( int, DeleteType = NoDelete );
 virtual void detach( Object _FAR &, DeleteType = NoDelete );
 void setData( int Indice, Object _FAR * pUnObjet)
 {  AbstractArray::setData(Indice, pUnObjet) ; }
void  AfficherNomsDomsDsBoite(TDialog*, int);
void AfficherNomsPropsDsBoiteListe(TDialog*, int);
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
        static PTStreamable build() ;
```

```
        virtual Pvoid read(Ripstream ) ;
        virtual void write(Ropstream ) ;

} ;

//*****************************************************************
//*****************************************************************
//                NOM DE LA CLASSE:TTableauProprietesDsDom
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-----------------------------------------------------------------------
class TCoupleIndiceDsTabEtBoolOpt ;
class  TTableauProprietesDsDom  : public TArrayWithHolesOfLinks
{

public :
TTableauProprietesDsDom(TConteneurLiens * pContDeArray,
                                                        int upper,
                                                        int lower = 0, sizeType aDelta = 0 )
                : TArrayWithHolesOfLinks( upper, lower, aDelta )
{}
TTableauProprietesDsDom(int upper, int lower = 0, sizeType aDelta = 0 )
                        :TArrayWithHolesOfLinks( upper, lower, aDelta )
        { }
BOOL PropDsDomEstMembre(TCoupleIndiceDsTabEtBoolOpt* ,char *,char*,HWND) ;
//<< >> << >> Methode de gestion des flux
        virtual Pvoid read(Ripstream ) ;
        virtual void write(Ropstream ) ;

};
//*****************************************************************
//*****************************************************************
//                NOM DE LA CLASSE:TTableauDomaines
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants.
//
//Classes amies:
//
//Operateurs surcharges:
//-----------------------------------------------------------------------
```

```
class  TTableauDomaines  : public TArrayWithHolesOfLinks
{

public :
TTableauDomaines(TConteneurLiens * pContDeArray,
                                                        int upper,
                                                        int lower = 0, sizeType aDelta = 0 )
                : TArrayWithHolesOfLinks( upper, lower, aDelta )
{ }
TTableauDomaines(int upper, int lower = 0, sizeType aDelta = 0 )
                        :TArrayWithHolesOfLinks( upper, lower, aDelta )
        { }

//<< >> << >> Methode de gestion des flux
        virtual Pvoid read(Ripstream ) ;
        virtual void write(Ropstream ) ;

};




//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TConteneurLiensHERIT
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------

_CLASSDEF(TConteneurLiensHERIT)
class    TConteneurLiensHERIT : public TConteneurLiens
{
friend TConteneurLiensHERITCOMPOS:
//                                  ----------PRIVE----------

//->->->->->->->->->->-> Liste des methodes

//<< >> << >> Methode de gestion des flux


//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
public :
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur

~
```

```
TConteneurLiensHERIT();

//[]->Methodes d'information
BOOL    LienEstMembre(TSchemaClasse * , TSchemaClasse *,HWND) ;
BOOL     IsPropValeurDsClasse(RTMessage ,TSchemaClasse*) ;
//->[]Methodes de manipulation
void DetruireClasseEtDependances(LPSTR,HWND) ;
void AddLinkSansToucherDicoEntree(LPSTR,LPSTR
                                                    ,HWND,Ripstream,int) ;
void SupprimerLienHeritage(LPSTR,LPSTR,HWND) ;
void VisualiserSuperClasses(HDC,unsigned int,int,TSchemaClasse *) ;
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux


} ;
//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TConteneurLiensCOMPOS
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------
class TLienCompositionConceptuel ;
_CLASSDEF(TConteneurLiensCOMPOS)
class    TConteneurLiensCOMPOS : public TConteneurLiens
{
friend TConteneurLiensHERITCOMPOS;

public :
//on maintient une liste de liens de composition créés  à partir d'une
//TFenSchemaClasse pour pouvoir leur donner des points origine et
//extrémités de leur lien graphique associé dans TFenGrapheStatique
TListeDeLiens*  pLiensComposDeTFenSchema;
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur

TConteneurLiensCOMPOS();
//[]->Methodes d'information
BOOL    LienEstMembre(TSchemaClasse * , TSchemaClasse *,LPSTR,HWND) ;
  //largeur totale des liens de compsoition de la classe en argument,
  // sert dans TConteneurLiensCOMPOS::VisualiserPropsValDsClasse
  //pour la visualisation dans une TFenSchemaClasse des liens de compsoition
  //en progressant HORIZONTALEMENT à chaque nouvelle classe à afficher
BOOL     IsPropValeurDsClasse(RTMessage ,TSchemaClasse*) ;
```

```
int LargeurTotalePropValDsClasse(HDC, LPSTR, HWND);
//->[]Methodes de manipulation
void DetruireClasseEtDependances(LPSTR,HWND) ;
void AjouterLienComposition(TLienCompositionConceptuel*,HWND);
void AddLinkSansToucherDicoEntree(LPSTR,LPSTR
                                                    ,HWND,Ripstream,int) ;
void AjouterLienCdeTFenSCh(TLienCompositionConceptuel*);
//donner une représentation graphique par défaut des liens crées à partir
//des TFenChemaClasse
void SetDefaultsGrLinksDeTFenSch(HDC) ;
void SetDefautsGrForOneCLink(TLienConceptuel *,HDC) ;

void SupprimerLienComposition(TLienConceptuel*,HWND) ;
void VisualiserPropsValDsClasse(HDC,unsigned int,int,TSchemaClasse *) ;
        //affiche dans la fenêtre schéma de classe, des icônes de classes
        //qui sont des propriétés à valeur dans classe pour la classe
        //courante en argument de la fonction
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux


} ;



//**************************************************************************
//**************************************************************************
//                NOM DE LA CLASSE: TConteneurLiensHERITCOMPOS
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//--------------------------------------------------------------------------

_CLASSDEF(TConteneurLiensHERITCOMPOS)
class    TConteneurLiensHERITCOMPOS : public TConteneurLiensHERIT


{

public :

//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur

TConteneurLiensHERITCOMPOS();
~TConteneurLiensHERITCOMPOS();
```

```
//[]->Methodes d'information

//->[]Methodes de manipulation
                                        //appelée par le constructeur
void CopierConteneurLiensHeritages();
                                        //appelée par le constructeur
void CopierConteneurLiensCompositions();

                                        //appelée par le destructeur
void DetruireCopieLiensHeritages();
                                        //appelée par le destructeur
void DetruireCopieLiensCompositions();

//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux

};


//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TConteneurLiensREFER
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------

_CLASSDEF(TConteneurLiensREFER)
//class  TDicoEntree ;
//class  TArrayWithHolesOfLinks ;
//class  TNomEtListeDoubleAssoc ;
class TLienReferenceConceptuel ;
class    TConteneurLiensREFER : public TConteneurLiens
{

public :
//on maintient une liste de liens de références créés  à partir d'une
//TFenSchemaClasse pour pouvoir leur donner des points origine et
//extrémités de leur lien graphique associé dans TFenGrapheStatique
TListeDeLiens*  pLiensReferDeTFenSchema;
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur

TConteneurLiensREFER();
//[]->Methodes d'information
```

```
BOOL    LienEstMembre(TSchemaClasse * , TSchemaClasse *,LPSTR,HWND) ;

  //largeur totale des liens de référence de la classe en argument,
  // sert dans TConteneurLiensREFER::VisualiserPropsValDsClasse
  //pour la visualisation dans une TFenSchemaClasse des liens de référence
  //en progressant HORIZONTALEMENT à chaque nouvelle classe à afficher
BOOL     IsPropValeurDsClasse(RTMessage ,TSchemaClasse*) ;
int LargeurTotalePropValDsClasse(HDC, LPSTR, HWND);
//->[]Methodes de manipulation
void DetruireClasseEtDependances(LPSTR,HWND) ;
void AjouterLienReference(TLienReferenceConceptuel*,HWND);
void AddLinkSansToucherDicoEntree(LPSTR,LPSTR,HWND,Ripstream,int) ;
void AjouterLienRdeTFenSCh(TLienReferenceConceptuel*);
//donner une représentation graphique par défaut des liens crées à partir
//des TFenChemaClasse
void SetDefaultsGrLinksDeTFenSch(HDC) ;
void SetDefautsGrForOneRLink(TLienConceptuel *,HDC) ;
void SupprimerLienReference(TLienConceptuel*,HWND) ;
void VisualiserPropsValDsClasse(HDC,unsigned int,int,TSchemaClasse *) ;
        //affiche dans la fenêtre schéma de classe, des icônes de classes
        //qui sont des propriétés à valeur dans classe pour la classe
        //courante en argument de la fonction
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux

} ;
//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TEnsembleConcept
//
//Description:
//
//Parametres de creation:
//
//Comportement du destructeur:
//
//Liste des ancetres:
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------

_CLASSDEF(TEnsembleConcept)
class  TConcept ;
class    TEnsembleConcept : public Set
{


public :

//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur


//[]->Methodes d'information
BOOL IdentifierConceptGr(POINT  posSouris,
```

```
                        TConcept * & pConceptIdentifie);

//->[]Methodes de manipulation
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux

//{{{{{{{{{{{ Methodes de reponses a un evenement

//^^^^^^^Surcharge d'operateur

} ;

//*************************************************************************
//*************************************************************************
//                NOM DE LA CLASSE: TAttribEtObjetAttribAssoc
//
//Description:
//
//Parametres de creation:
//Object & cleNomClasse :   clé de l' association objet de type String
//Object & valeurObjetClasse : valeur de l'association un objet de
//TSchemaClasse
//Comportement du destructeur
//
//Liste des ancetres: Association, TStreamable, Object, TShouldDelete
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------



_CLASSDEF(TAttribEtObjetAttribAssoc)
class    TAttribEtObjetAttribAssoc: public Association, public TStreamable

{



//                              ----------PRIVE----------

//->->->->->->->->->->-> Liste des methodes
//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const ;

        TAttribEtObjetAttribAssoc(StreamableInit)
        :Association(NOOBJECT,NOOBJECT) {}

//                              ++++++++++++++++++PUBLIQUE++++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
```

```
//+++++----- Constructeurs et destructeur
TAttribEtObjetAttribAssoc(Object & cleNomAttribut,Object & valeurObjetAttrib )
 : Association(cleNomAttribut, valeurObjetAttrib )
                    {
                    }


//[]->Methodes d'information

//->[]Methodes de manipulation

//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
          static PTStreamable build() {}
          virtual Pvoid read(Ripstream ) {}
          virtual void write(Ropstream ) {}

};

//*************************************************************************
//*************************************************************************
//                  NOM DE LA CLASSE: TDicoAttributs
//
//Description:
//TDicoAttributs est une classe conteneur contenant un ensemble
//d' associations instances de la classe TAttribEtObjetAttribAssoc
// chaque objet TAttribEtObjetAttribAssoc est un couple nom de l'attribut
// (clé de l' association) et un objet attribut (valeur de l'
//association) qui peut être un TlienCompositionConceptuel,
// un TLienReferenceConceptuel ou un TCoupleIndiceDsTabEtBoolOpt
//
//Parametres de creation:
//unsigned sz  taille de départ du dictionnaire
//
//Comportement du destructeur:
//
//Liste des ancetres:Dictionary,Set,Bag,Collection,Container,Object
//,TShouldDelete, TStreamable
//
//Liste des enfants:
//
//Classes amies:
//
//Operateurs surcharges:
//-------------------------------------------------------------------------



_CLASSDEF(TDicoAttributs)
class    TDicoAttributs: public TStreamable ,public Dictionary

{
 //                                  ------------------PRIVE----------------

//->->->->->->->->->->->-> Liste des methodes

//<< >> << >> Methode de gestion des flux

virtual const Pchar streamableName() const {}
```

```
        TDicoAttributs(StreamableInit) {}

//                              ++++++++++++++++++PUBLIQUE+++++++++++++++
public:
//@ @ @ @ @ @ @ @ @ @  Liste des donnees


//->->->->->->->->->->-> Liste des methodes
//+++++----- Constructeurs et destructeur
TDicoAttributs( unsigned sz = DEFAULT_HASH_TABLE_SIZE ) :       Dictionary(sz)
      {
        }

//[]->Methodes d'information

//->[]Methodes de manipulation

//void ReconstitueLiensHeritage() ;
//OK<----Methodes d'autorisation

//<< >> << >> Methode de gestion des flux
        static PTStreamable build() {}
        virtual void write(Ropstream ) {}
        virtual Pvoid read(Ripstream ) {}

};
#endif




//=============================================================================
//              NOM DE FICHIER: CON_TDA.CPP
//
//
//Liste des classes:
//                      -TDicoClasses
//                              -TDicoEntree
//                              -TConteneurLiens
//                              -TConteneurLiensHERIT
//                              -TConteneurLiensCOMPOS
//                              -TConteneurLiensREFER
//                      -TNomEtObjetclasseAssoc
//                      -TNomEtListeDoubleAssoc
//                      -TDeuxListsOrigExtr
//                              -TArrayWithHolesOfLinks
//                              -TTableauProprietesDsDom
//                              -TConteneurLiensHERITCOMPOS
//                              -TEnsembleConcept
//=============================================================================
//*********Remarques

//*********Liste des fichiers inclus
#include "CON_TDA.HPP"
```

```
#include "entier.hpp"
//=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=
//déclaration de VARIABLES GLOBALES définies ailleurs
extern TDicoClasses                    * DicoClasses ;
extern TConteneurLiensHERIT  * ConteneurLiensHERIT;
extern TConteneurLiensCOMPOS        * ConteneurLiensCOMPOS ;
extern TConteneurLiensREFER          * ConteneurLiensREFER ;
extern TTableauProprietesDsDom       * ConteneurProprietes;
extern TTableauDomaines              * ConteneurDomaines ;

//=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=
//déclaration de FONCTIONS INDEPENDANTES
void extraire (char *, int, int) ;
void lire(char *, int, Ripstream );
void ecrire(char *,int , Ropstream ) ;
void ecrireTexte (char *, Ropstream) ;
//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE:TDicoClasses
//*************************************************************************
//-------------
//>>         NOM DE LA METHODE: constructeur
//
TDicoClasses::TDicoClasses(StreamableInit)
{
}
 //-------------
//*************************************************************************
//*************************************************************************
//              METHODE NOM MEMBRE DE CLASSE (INDEPENDANTE) :
//*************************************************************************
// OFFRANT DES SERVICES A TDicoClasse
//
//         NOM DE LA METHODE: EcrireUneClasseDsBteListe
//
//         -
//Methodes appelees: EcrireUneClasseDsBteListe de TNometObjetclassseAssoc
//forEach
//Commentaires:
//
//
//
void      EcrireUneClasseDsBteListe ( Object & UneAssoc, void * plb)
{
TSchemaClasse * psch =
 & ((TSchemaClasse&) ( ((TNomEtObjetclasseAssoc&) UneAssoc).value() ) ) ;

          ((TListBox*)plb)->AddString(psch->NomClasse) ;
}

//-------------
//         NOM DE LA METHODE:TDicoClasses:: EcrireClassesDsBoiteListe
//
         -
//Methodes appelees: EcrireUneClasseDsBteListe de TNometObjetclassseAssoc
//forEach
//Commentaires:
//met en oeuvre un itérateur interne (forEach) pour effectuer
//une même action ici EcrireUneClasseDsBteListe pour écrire les noms des
//classes dans la boîte liste de droite
```

```
//
//
void TDicoClasses::EcrireClassesDsBoiteListe(TListBox          * plBox)
{
 forEach(EcrireUneClasseDsBteListe , plBox ) ;
}
//-------------
//        NOM DE LA METHODE: TDicoClasses::AjouterNouvelleEntree
//
        -
//Methodes appelees: EcrireUneClasseDsBteListe de TNometObjetclassseAssoc
//Commentaires:
//
//
//
void TDicoClasses::AjouterNouvelleEntree(TSchemaClasse  *pSch)
{//début fonction
        String    *str =  new String(pSch->NomClasse);
        //pSch = new   TSchemaClasse (szNom) ;
   TNomEtObjetclasseAssoc *entree =
                                                new TNomEtObjetclasseAssoc( *str, *pSch),
        add(*entree ) ;

  //à chaque ajout d'une assoc ayant comme value() une TSchemaClasse
  //dans DicoClasses
  //je crée une assoc (nomDeClasse, pairedDeListeOriginesEtExtrémitéVide)
  //dans le DicoEntree de ConteneurLiensHERIT et de ConteneurLiensCOMPOS
        ConteneurLiensHERIT->AjouterAssocDsDicoEntree(pSch->NomClasse);
        ConteneurLiensCOMPOS->AjouterAssocDsDicoEntree(pSch->NomClasse);
        ConteneurLiensREFER->AjouterAssocDsDicoEntree(pSch->NomClasse);
}//fin fonction

//-------------
//        NOM DE LA METHODE: TDicoClasses::EstMembre (version 1)
//
//Methodes appelees: EcrireUneClasseDsBteListe de TNometObjetclassseAssoc
//forEach
//



BOOL TDicoClasses::EstMembre(LPSTR szNomCl,TSchemaClasse* & pSchema)
{

String  * pstr = new String (szNomCl) ;
TNomEtObjetclasseAssoc & rAssoc =
                (TNomEtObjetclasseAssoc &) lookup(*pstr) ;
        if( rAssoc== NOOBJECT )
     {
                pSchema = new   TSchemaClasse(szNomCl);
                return 0;
               }
 else
  {
                pSchema =(TSchemaClasse *) & (rAssoc.value() ) ;
                return 1;
  }

}
```

```
//-------------
//        NOM DE LA METHODE: TDicoClasses::EstMembre (version 2)
//Methodes appelees:
//forEach
//Commentaires:
//
//
BOOL TDicoClasses::EstMembre(LPSTR szNomConfirm,
                                                    TSchemaClasse* & pSchemaConfirm ,

        BOOL & bHasBeenCreated,

HWND hwinConfirm)
{//début fonction

String  * pstrConfirm = new String (szNomConfirm) ;
TNomEtObjetclasseAssoc & rAssoc =
                (TNomEtObjetclasseAssoc &) lookup(*pstrConfirm) ;
        if( rAssoc== NOOBJECT )
                {//début if (rassoc==
                 char bufinfo[150]; strcpy(bufinfo,"la classe ");
                 strcat(bufinfo,szNomConfirm) ;
                 strcat(bufinfo,
                   " n'existe pas dans le dictionnaire de classes ");
                 MessageBeep(MB_ICONINFORMATION);
                 MessageBox(hwinConfirm,bufinfo,"Message O Etoile",

        MB_OK|MB_ICONINFORMATION) ;
                 char bufquestion[150];
                 strcpy(bufquestion,"Voulez-vous créer la classe ");
                 strcat(bufquestion,szNomConfirm) ;
                 int res = MessageBox(hwinConfirm,bufquestion,"Message",

MB_ICONQUESTION|MB_YESNOCANCEL) ;
                 if( res==IDYES)
                  {//début if res ==
                   bHasBeenCreated = 1 ;
                   pSchemaConfirm = new   TSchemaClasse(szNomConfirm);
                   //ajouter la nouvelle classe dans le dictionnaire des classes
                   TNomEtObjetclasseAssoc *entreeConfirm =
                          new TNomEtObjetclasseAssoc( *pstrConfirm, *pSchemaConfirm);
                   add(*entreeConfirm ) ;

                   TFenPrinc *pFenPrinc =
                            (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
                  //informer la fenêtre principale qu'il y a des modifications
                  //de la base initiale; donc ne peut pas quitter ou fermer la
                  //base sans prévenir le développeur la sauvegarde éventuelle
      //de ces modifications
                  pFenPrinc->SetIsDirty(TRUE);

                  //à chaque ajout d'une assoc ayant comme value() une
                  // TSchemaClasse dans DicoClasses je crée une assoc
                  //(nomDeClasse, pairedDeListeOriginesEtExtrémitéVide)
                  //dans le DicoEntree de ConteneurLiensHERIT , de
                  //ConteneurLiensCOMPOS et ConteneurLiensREFER
                  ConteneurLiensHERIT->
                                        AjouterAssocDsDicoEntree(szNomConfirm);
                  ConteneurLiensCOMPOS->
                                        AjouterAssocDsDicoEntree(szNomConfirm);
```

```
                    ConteneurLiensREFER->
                                        AjouterAssocDsDicoEntree(szNomConfirm);
                    }//fin if res ==
                    else
                      bHasBeenCreated = 0 ;
                   return 0;
                  }//fin de if rassoc ==
 else
  {//début else
                  pSchemaConfirm =(TSchemaClasse *) & (rAssoc.value() ) ;
                  return 1;
  }//fin else

}//fin fonction

//-------------
//          NOM DE LA METHODE: TDicoClasses::IdentifierIconeClasse
//

TSchemaClasse  * TDicoClasses::IdentifierIconeClasse
                                    (RTMessage Msg, RECT & refRectangleDeDefilement )
{//début fonction
TFenPrinc *pFenPrinc =
        (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
TScroller * unDefile =  pFenPrinc->GrapheStatique->
                                     FenGrapheStatique->Scroller ;

POINT ptCurseur ;
ptCurseur = MAKEPOINT(Msg.LParam) ;

RContainerIterator identifieIterator = initIterator();

 while ( int(identifieIterator) != 0 )
 {//début while
        RTNomEtObjetclasseAssoc identifieObject =
                        (RTNomEtObjetclasseAssoc) identifieIterator++;
                PTSchemaClasse  pSchema    =
                        (PTSchemaClasse) &(identifieObject.value());


                refRectangleDeDefilement =
                    pSchema->FenSchemaClasse->rIconeClasse.Forme ;
                refRectangleDeDefilement.top -=
                        (unDefile->YPos)*(unDefile->YUnit) ;
                refRectangleDeDefilement.bottom -=
                        (unDefile->YPos)*(unDefile->YUnit) ;
                refRectangleDeDefilement.left -=
                        (unDefile->XPos)*(unDefile->XUnit) ;
                refRectangleDeDefilement.right -=
                        (unDefile->XPos)*(unDefile->XUnit) ;


                if( PtInRect(&refRectangleDeDefilement,
                //(pSchema->FenSchemaClasse->rIconeClasse.Forme)
                                        ptCurseur) )
                 {//début if
                  delete &identifieIterator;
                  return pSchema ;
                 }//fin if PtInRect
 }//fin while
```

```
 delete &identifielterator;
 return NULL ;
}//fin fonction

//-------------
//          NOM DE LA METHODE: TDicoClasses::DetacherClasse
//
void TDicoClasses::DetacherClasse(LPSTR NomCl)
{
String  * pstr = new String (NomCl) ;
TNomEtObjetclasseAssoc & rAssoc =
                    (TNomEtObjetclasseAssoc &) lookup(*pstr) ;
          if( rAssoc== NOOBJECT )
                    {
                     char bufdetach[180] ; strcpy(bufdetach,NomCl);
                     strcat(bufdetach,
                     "  non trouvée dans le dictionnaire des classes ");
                     MessageBeep(MB_ICONEXCLAMATION);
                     MessageBox(GetApplicationObject()->MainWindow->HWindow,
                       bufdetach,"O Etoile Message ",MB_ICONEXCLAMATION|MB_OK) ;
                    }
 else
  {
  //enlève l'assoc du conteneur mais ne la détruit pas !!
  ownsElements(0);
  detach(rAssoc ) ;
  }


}
//[]<- Methodes de manipulation
//-------------
//<-          NOM DE LA METHODE:TDicoClasses::DetruireClasseEtDependances
//

void TDicoClasses::DetruireClasseEtDependances(LPSTR szClassetoDestroy,
                                    int IndexRightListBox)
{//début

TFenPrinc *pFenPrinc =
          (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
String  * pstr = new String (szClassetoDestroy) ;
TNomEtObjetclasseAssoc & rAssoctoDestroy =
                    (TNomEtObjetclasseAssoc &) lookup(*pstr) ;
          if( rAssoctoDestroy== NOOBJECT )
                    {//début si
                     char bufdetach[180] ; strcpy(bufdetach,szClassetoDestroy);
                     strcat(bufdetach,
                     "  non trouvée dans le dictionnaire des classes ");
                     MessageBeep(MB_ICONEXCLAMATION);
                     MessageBox(pFenPrinc->HWindow,
                       bufdetach,"O Etoile Message ",MB_ICONEXCLAMATION|MB_OK) ;
                    }//fin si
 else
  {//début else

  //enlève l'assoc du dictionnaire de classe mais ne la détruit pas !!
  ownsElements(0);
  detach(rAssoctoDestroy ) ;
  //enlever dans la boite liste de droite le nom de la classe
```

```
  //que l'on supprime
        pFenPrinc->DeleteStrngInRightLstBox(IndexRightListBox) ;
  //on supprime l'objet interface et l'élément d'interface visuel
  //associé
  ((TSchemaClasse &)rAssoctoDestroy.value()).
                        FenSchemaClasse->ShutDownWindow();
  //demander au conteneur de liens de supprimer tout ce qui a rapport
  //dans le conteneur de liens avec la classe qu'on vient d'éliminer
  ConteneurLiensHERIT->
    DetruireClasseEtDependances(szClassetoDestroy,pFenPrinc->HWindow);
  ConteneurLiensCOMPOS->
        DetruireClasseEtDependances(szClassetoDestroy,pFenPrinc->HWindow);
  ConteneurLiensREFER->
        DetruireClasseEtDependances(szClassetoDestroy,pFenPrinc->HWindow);
  //penser à supprimer ici l'objet TSchemaClasse lui même
  //auparavant la munir d'un destructeur

  }//fin else


}//fin fonction

//-------------
//        NOM DE LA METHODE: TDicoClasses::RenommerClasse
//
void TDicoClasses::RenommerClasse(LPSTR AncNom,
                                                        LPSTR NouvNom)
{

  TSchemaClasse       * pSchema ;
  //copie locale de l' ancien nom (qui est en fait le champ NomClasse
  //du RSchemaClasse à renommer et donc sera écrasé par le nouveau
  //nom quand l'appel vient de TFenGrapheStatique::RenommerClasse)
  //sinon on risque de passer pour les deux arguments de la fonction
  //TConteneurLiens::RenommerClasse la même chaine de caractère
  //c'est à dire le nouveau nom
  char *   szAncienNomLocal = new char [80];
  strcpy(szAncienNomLocal,AncNom) ;
 if( EstMembre(AncNom,pSchema) )
        {
        DetacherClasse (szAncienNomLocal) ;
        strcpy(pSchema->NomClasse, NouvNom ) ;
        //écrit le nom de la classe dans la barre de titre
        char bufrenom[LGMAXEXPR] ; strcpy(bufrenom,"Classe - ");
        strcat(bufrenom,pSchema->NomClasse);
        pSchema->SetCaption(bufrenom);

        String * pstr = new String( NouvNom );
        TNomEtObjetclasseAssoc *entree =
                                        new TNomEtObjetclasseAssoc( *pstr, *pSchema);
        add(*entree );
    //mise à jour des dictionnaires de liens HERIT,COMPOS
        ConteneurLiensHERIT->RenommerClasse(szAncienNomLocal,NouvNom);
        ConteneurLiensCOMPOS->RenommerClasse(szAncienNomLocal,NouvNom);
        ConteneurLiensREFER->RenommerClasse(szAncienNomLocal,NouvNom);
        //on demande au conteneur de liens de maintenir une liste de classe
        //qui viennent d'être renommées pour ensuite lors de l'affichage
        //de la zone client de TFenGrapheStatique REAJUSTER les points
        //origines et extrémités des liens aboutissant sur la classe
        //qui est l'objet du renommage (cela permet d'éviter des liens qui pendent)
```

```
        ConteneurLiensHERIT->AddClassJustRenamedInList(pSchema ) ;
        ConteneurLiensCOMPOS->AddClassJustRenamedInList(pSchema ) ;
        ConteneurLiensREFER->AddClassJustRenamedInList(pSchema ) ;
        delete [] szAncienNomLocal ;
        }
}


//<< >> << >> << >>Methode de gestion des flux
//-------------

//>>        NOM DE LA METHODE:TDicoClasses::write
//

void TDicoClasses::write(Ropstream os)
{// début fonction TDicoClasses::write(Ropstream os)
        RContainerIterator writeIterator = initIterator();
        ecrire ("TDicoClasse - Nombre de classes    :    %3d\r\n",
                        getItemsInContainer(),
        os
        ) ;

        while ( int(writeIterator) != 0 )
        {
        RObject writeObject = writeIterator++;
        if ( writeObject != NOOBJECT )
                ((PTNomEtObjetclasseAssoc )(&writeObject))->write (os);
        }
        delete &writeIterator;
}// fin fonction TDicoClasses::write(Ropstream os)

//-------------
//>>        NOM DE LA METHODE:TDicoClasses::read
//
Pvoid TDicoClasses::read(Ripstream is)
{
countType       NumAssocs ;
char            szNumAssocs [4] ;
int                     nLongueur = strlen(
                                "TDicoClasse - Nombre de classes    :    %3d\r\n"
                         ) ;
char            * szTampon = new char [nLongueur + 1] ;
int             NbreCar = 3 ;// correspond à %3d
int             PosInit = nLongueur - NbreCar - 2 ;

        lire            (szTampon, nLongueur, is) ;
        extraire        (szTampon, PosInit, NbreCar) ;
   strcpy               (szNumAssocs, szTampon) ;

        NumAssocs = atoi ( szNumAssocs ) ;

        for (int i = 0 ; i < NumAssocs ; ++i )
        {//début for
   String   *str =  new String("");
        TSchemaClasse  *pSch = new   TSchemaClasse ("") ;
        PTNomEtObjetclasseAssoc entree =
                        new TNomEtObjetclasseAssoc( *str, *pSch );
        entree->read(is);
   add(* (entree));  //add met à jour itemsInContainer
   //à chaque ajout d'une assoc ayant comme value() une TSchemaClasse
```

```
        //dans DicoClasses
        //je crée une assoc (nomDeClasse, pairedDeListeOriginesEtExtrémitéVide)
        //dans le DicoEntree de ConteneurLiens  et de ConteneurLiensCOMPOS

        ConteneurLiensHERIT->
                AjouterAssocDsDicoEntree(
          (
                (TSchemaClasse&)((*entree).value())
          ).NomClasse
                                                );
        ConteneurLiensCOMPOS->
                AjouterAssocDsDicoEntree(
        (
                (TSchemaClasse&)((*entree).value())
        ).NomClasse
                                                );

        ConteneurLiensREFER->
                AjouterAssocDsDicoEntree(
        (
                (TSchemaClasse&)((*entree).value())
        ).NomClasse
                                                );

        }//fin for
delete szTampon ;
return this;
}// fin de fonction

//************************************************************************
//************************************************************************
//              METHODE NON MEMBRE DE CLASSE (INDEPENDANTE) :
//************************************************************************
// OFFRANT DES SERVICES A TDicoClasse
//
//        NOM DE LA METHODE: ReconstLiensHeritage
//
        -
//Methodes appelees: ReconstitueLiensHeritage de TSchemaClasse
//forEach
//Commentaires:

//[]<- Methodes de manipulation
//-------------

//-------------
//>>        NOM DE LA METHODE: streamableName
//
const Pchar TDicoClasses::streamableName() const
{
return "TDicoClasses" ;
}

//                              ++++++++++++++++++PUBLIQUE+++++++++++++++


//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>        NOM DE LA METHODE: build
//
```

```
PTStreamable TDicoClasses::build()
{
 return new TDicoClasses(streamableInit) ;
}

//*********************************************************************
//*********************************************************************
//              METHODES DE LA CLASSE:TNomEtObjetclasseAssoc
//*********************************************************************


//                                   ------------------PRIVE--------------
//<< >> << >> << >>Methode de gestion des flux
//>>        NOM DE LA METHODE: constructeur
//
TNomEtObjetclasseAssoc::TNomEtObjetclasseAssoc(StreamableInit)
  :Association(NOOBJECT,NOOBJECT)
{
}
//
//-------------
//>>        NOM DE LA METHODE: write
//
void TNomEtObjetclasseAssoc::write(Ropstream os)
{
 ((TSchemaClasse & ) value())   write ( os );

}

//-------------
//>>        NOM DE LA METHODE: read
//
Pvoid TNomEtObjetclasseAssoc::read(Ripstream is)
{

        ((TSchemaClasse & ) value()) . read ( is ) .

   // l'affectation ci dessous doit être permise par la surdéfinition
        // dans la classe String suivante
        // String& operator = ( const String _FAR & );
        // et le constructeur suivant à un argument
        // String( const char _FAR * = "" );
        // qui réalise la conversion automatique d'un char * en String
        (( String & )  key()) = ((TSchemaClasse & ) value()).NomClasse ;

   return this;
}
//-------------
//>>        NOM DE LA METHODE: streamableName
//
const Pchar TNomEtObjetclasseAssoc::streamableName() const
{
return "TNomEtObjetclasseAssoc" ;
}

//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++


//<< >> << >> << >>Methode de gestion des flux
//-------------
```

```
//>>          NOM DE LA METHODE: build
//
PTStreamable TNomEtObjetclasseAssoc::build()
{
 return new TNomEtObjetclasseAssoc(streamableInit) ;
}

void TNomEtObjetclasseAssoc::RenommerClasse(LPSTR NewAppellation)
{

 ( (TSchemaClasse &) value() ). RenommerClasse(NewAppellation) ;
   (( String & )  key()) = ((TSchemaClasse & ) value()).NomClasse ;
}

//*************************************************************************
//*************************************************************************
//                METHODES DE LA CLASSE: TConteneurLiens
//*************************************************************************

//                                        ----------PRIVE----------
//[]<- Methodes de manipulation
//-------------
//<-          NOM DE LA METHODE:
// Concat_ListeOrig_ListeExtr(TListeDeLiens * plisteLnOrigs,
//                                                          TListeDeLiens * plisteLnExtrs)
//
TListeDeLiens* TConteneurLiens::
  Concat_ListeOrig_ListeExtr(TListeDeLiens * plisteLnOrigs,
                                                          TListeDeLiens * plisteLnExtrs)
{//début fonction
TListeDeLiens* plisteliens= new TListeDeLiens();

//Parcourir la liste  des liens dont la classe est origine
RContainerIterator rListeORIGIterator =plisteLnOrigs->initIterator();
//Accéder au lien par l'intermédiaire de l'itérateur
// et ajouter le lien dans la liste de liens
while ( int(rListeORIGIterator) != 0 )
 {//début while
 TLienConceptuel & rUnLienConceptuel =
     (TLienConceptuel &)rListeORIGIterator++;
         if ( rUnLienConceptuel != NOOBJECT )
         {//début if

           plisteliens->add(rUnLienConceptuel);
         }//fin if
 }//fin while
 delete &rListeORIGIterator;


//Parcourir la liste  des liens dont la classe est extrémité
RContainerIterator rListeEXTRIterator =plisteLnExtrs->initIterator();
//Accéder au lien par l'intermédiaire de l'itérateur
// et ajouter le lien dans la liste de liens
while ( int(rListeEXTRIterator) != 0 )
 {//début while
 TLienConceptuel & rUnLienConceptuelExtr =
           (TLienConceptuel &)rListeEXTRIterator++;
         if ( rUnLienConceptuelExtr != NOOBJECT )
         {//début if
```

```
            plisteliens->add(rUnLienConceptuelExtr);
          }//fin if
  }//fin while
  delete &rListeEXTRIterator;

return plisteliens;
}//fin fonction



//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>          NOM DE LA METHODE:
//     TConteneurLiens::TConteneurLiens(StreamableInit)
//
//Parametres:
//Renvoie:
//Objets ou Donnees utilisees:
//Methodes appelees:
//Commentaires:
//
//
TConteneurLiens::TConteneurLiens(StreamableInit)
{//début fonction

pdicoEntree =new TDicoEntree() ;
ptableauliens =new TArrayWithHolesOfLinks(this,10,0,10 ) ;

//on maintient une liste de liens d'héritage créés  à partir d'une
//TFenSchemaClasse pour pouvoir leur donner des points origine et
//extrémités de leur lien graphique associé dans TFenGrapheStatique
pLiensHeritDeTFenSchemaClasse = new TListeDeLiens();
//on demande au conteneur de liens de maintenir une liste de classe
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage (cela permet des liens qui pendent)
pClassesJustBeenRenamed  = new TListeDeClasses() ;
}//fin fonction

//>>          NOM DE LA METHODE:TConteneurLiens::streamableName()
const Pchar TConteneurLiens::streamableName() const
{//début fonction

return "TConteneurLiens" ,

}//fin fonction

//                                    ++++++++++++++++++PUBLIQUE+++++++++++++++
//+++++----- Constructeurs et destructeur
TConteneurLiens::TConteneurLiens()
{
pdicoEntree =new TDicoEntree() ;
ptableauliens = new TArrayWithHolesOfLinks(this,10,0,10) ;

//on maintient une liste de liens d'héritage créés  à partir d'une
//TFenSchemaClasse pour pouvoir leur donner des points origine et
//extrémités de leur lien graphique associé dans TFenGrapheStatique
pLiensHeritDeTFenSchemaClasse = new TListeDeLiens();
//on demande au conteneur de liens de maintenir une liste de classe
```

```
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage (cela permet des liens qui pendent)
pClassesJustBeenRenamed  = new TListeDeClasses() ;
}
//[]-> Methodes d'information

//->        NOM DE LA METHODE:
//TConteneurLiens::IdentifierLien
TLienConceptuel         *         TConteneurLiens::
                                                IdentifierLien(RTMessage Msg)
{//début fonction

RContainerIterator identifieIterator = ptableauliens->initIterator();

 while ( int(identifieIterator) != 0 )
 {//début while

        RTLienConceptuel identifieAnObject =
                        (RTLienConceptuel)identifieIterator++;
        if ( identifieAnObject != NOOBJECT )
  {//début if

         TLienConceptuel       * pLienConceptObject =
                                                identifieAnObject.SuisJeClique(Msg) ;

         if(pLienConceptObject)  return pLienConceptObject ;

        }//fin if( identifieAnObject != NOOBJECT )
 }//fin while

 delete &identifieIterator;

 return NULL ;
}//fin fonction
//->        NOM DE LA METHODE:
//TConteneurLiens::LargeurTotaleSuperClasses
//
//Commentaires:
//largeur totale superclasses de la classe en argument, sert dans
//TSchemaClasse::VisualiserSuperClasses pour la visualisation dans une
//TFenSchemaClasse des superclasses en progressant HORIZONTALEMENT
//à chaque nouvelle classe à afficher

int TConteneurLiens::LargeurTotaleSuperClasses(
                                                HDC  hDCLargTot,
                                                LPSTR szNomClasseLargTot,
                                                HWND hwindLargTot)
{//début fonction

 int LargIconeCourante,LargeurTotaleSuperClasses ;

 TSchemaClasse         * pSchema ;
 //initialisation à zéro
 LargeurTotaleSuperClasses = 0;

 TListeDeLiens*  listeLiensTaille = ConteneurLiensHERIT->
         LiensDtClasseEstOrigine(szNomClasseLargTot,hwindLargTot);
 RContainerIterator listTailleIterator =listeLiensTaille ->initIterator();
```

```
while ( int(listTailleIterator) != 0 )
 {//début while
 TLienConceptuel & rlienconTaille =
            (TLienConceptuel &) listTailleIterator++;
        if ( rlienconTaille!= NOOBJECT )
        {//début if
         pSchema = rlienconTaille.ExtremiteLien();

        //pSchema->FenSchemaClasse->rIconeClasse.
        //            SetLargHautIcone(hDCLargTot,pSchema);
        HFONT pol1,pol_orig1;
        LOGFONT pol_log;

        pol_log.lfHeight = 16;
        pol_log.lfWidth =0;
        pol_log.lfItalic = 0 ;
        pol_log.lfWeight = 400 ; pol_log.lfStrikeOut=FALSE ;
        pol_log.lfUnderline = FALSE ;
        pol_log.lfCharSet = ANSI_CHARSET ;
        pol_log.lfEscapement = 0 ;pol_log.lfOrientation= 0;
        pol_log.lfOutPrecision=0;pol_log.lfClipPrecision = 0 ;
        //pol_log.lfQuality =PROOF_QUALITY ;
        pol_log.lfPitchAndFamily = FF_MODERN | FIXED_PITCH ;
        strcpy(pol_log.lfFaceName,"Courier New");

        pol1 = CreateFontIndirect(&pol_log);

        pol_orig1 = SelectObject(hDCLargTot, pol1);
        long Taille;
        Taille=GetTextExtent(hDCLargTot,
                        pSchema->NomClasse,strlen(pSchema->NomClasse) );

        SelectObject(hDCLargTot,pol_orig1);
        DeleteObject(pol1);

        LargIconeCourante=LOWORD(Taille) + 5;
        //LargIconeCourante =
        //      pSchema->FenSchemaClasse->rIconeClasse.Forme.right -
        //                  pSchema->FenSchemaClasse->rIconeClasse.Forme.left ;

        LargeurTotaleSuperClasses +=LargIconeCourante ;

        }//fin if ( rlienconTaille!= NOOBJECT )
 }//fin while
delete &listTailleIterator;

return  LargeurTotaleSuperClasses ;

}//fin fonction

//->        NOM DE LA METHODE:
//TConteneurLiens::SuperClassesDe(LPSTR szlienNomClasse,HWND hfene )

void TConteneurLiens::SuperClassesDe (TSetClasses  * psetCl,
                        TSchemaClasse *pSch, HWND hwind)
{ //début fonction

TListeDeLiens*  listeSuperClasses = LiensDtClasseEstOrigine
                                        (pSch->NomClasse,hwind);
RContainerIterator SuperClassesIterator =
```

```
                                    listeSuperClasses-> initIterator();
if (!int(SuperClassesIterator))
                    {//debut if
                    delete &SuperClassesIterator;
                    return ;
                    }//fin if (!SuperClassesIterator)

TSchemaClasse *psc;
while (int(SuperClassesIterator))
          {//début while
          TLienConceptuel & rliencon=
                    (TLienConceptuel &)SuperClassesIterator++;
            if ( rliencon != NOOBJECT )
                    {//début if
                      psc = rliencon.ExtremiteLien();
                      psetCl->add (*psc) ;
                SuperClassesDe (psetCl, psc, hwind);
                    }//fin if
          } // fin du while
}// fin fonction



//->          NOM DE LA METHODE:
//TConteneurLiens::LiensDtClasseEstOrigine(LPSTR szlienNomClasse,HWND hfene )

TListeDeLiens* TConteneurLiens::
          LiensDtClasseEstOrigine(LPSTR szlienNomClasse,HWND hfene )
{//début fonction

//demander au DicoEntree la liste des indices (origines) du tableau
//de liens pour la classe de nom  szlienNomClasse
//Convention : A hérite de B
//A est origine du lien  et B est extrémité du lien
TListeIndices  * plistIndsOrig =
                    pdicoEntree->ObtenirListeIndicesOrigines(szlienNomClasse,hfene);
//Créer dans la mémoire dynamique une liste de LIENS et non d'INDICES
TListeDeLiens* plistliensOrig = new  TListeDeLiens();

//Parcourir la liste  des INDICES
RContainerIterator listIndsIterator =plistIndsOrig-> initIterator();
//Accéder au lien par l'intermédiaire de l'indice
// et ajouter le lien dans la liste de liens
while ( int(listIndsIterator) != 0 )
 {//début while
 TEntier & rUnIndice= (TEntier &) listIndsIterator++;
          if ( rUnIndice != NOOBJECT )
          {//début if

            plistliensOrig->add((*ptableauliens)[rUnIndice]);
          }//fin if
 }//fin while
 delete &listIndsIterator;

return plistliensOrig ;

}//fin fonction

//-------------
//->          NOM DE LA METHODE
```

```
//LiensDtClasseEstExtremite(LPSTR szExtNomDeClasse ,HWND hwext)
//
//


TListeDeLiens* TConteneurLiens::
                    LiensDtClasseEstExtremite(LPSTR szExtNomDeClasse ,HWND hwext)
{//début fonction

//demander au DicoEntree la liste des indices (extrémités) du tableau
//de liens pour la classe de nom  szExtNomDeClasse
//Convention : A hérite de B
//A est origine du lien  et B est extrémité du lien
TListeIndices  * plistIndsExtr=
        pdicoEntree->ObtenirListeIndicesExtremites(szExtNomDeClasse,hwext);
//Créer dans la mémoire dynamique une liste de LIENS et non d'INDICES
TListeDeLiens* plistliensExtr= new  TListeDeLiens();

//Parcourir la liste  des INDICES
RContainerIterator listIndsIterator =plistIndsExtr-> initIterator();
//Accéder au lien par l'intermédiaire de l'indice
// et ajouter le lien dans la liste de liens
while ( int(listIndsIterator) != 0 )
 {//début while
 TEntier & rUnIndice= (TEntier &) listIndsIterator++;
        if ( rUnIndice != NOOBJECT )
        {//début if

          plistliensExtr->add((*ptableauliens)[rUnIndice]);
        }//fin if
 }//fin while
 delete &listIndsIterator;

return plistliensExtr;


}//fin fonction

//----------
//->          NOM DE LA METHODE:
//LiensDeLaClasse(LPSTR szInksNomDeClasse, HWND hwl)

TListeDeLiens* TConteneurLiens::
        LiensDeLaClasse(LPSTR szInksNomDeClasse, HWND hwl)
{//début fonction
//Demander la liste des liens dont la classe est origine
TListeDeLiens* pLiensOrigines =
         LiensDtClasseEstOrigine(szInksNomDeClasse,hwl) ;
//Demander la liste des liens dont la classe est extrémité
TListeDeLiens* pLiensExtremites =
                    LiensDtClasseEstExtremite(szInksNomDeClasse,hwl);
//Conacaténer les deux listes
TListeDeLiens* pLiensDeLaClasse =
                     Concat_ListeOrig_ListeExtr(pLiensOrigines,pLiensExtremites);
//renvoyer la liste résultante
return pLiensDeLaClasse ;

}//fin fonction

//[]<- Methodes de manipulation
```

```
//-------------
//<-        NOM DE LA METHODE
//  TConteneurLiens::RenommerClasse
//
//Parametres:LPSTR un nom de classe
//        -
:
void TConteneurLiens::RenommerClasse(LPSTR AncNomCon,LPSTR NouvNomCon)
{
pdicoEntree->RenommerClasse(AncNomCon,NouvNomCon)  ;
}

//<-        NOM DE LA METHODE:
//  TConteneurLiens::AjouterAssocDsDicoEntree
//
//Parametres:LPSTR un nom de classe
//        -
//Renvoie:
//Objets ou Donnees utilisees:                    (M) indique une modif de la donnee
//        -
//Objet crees:
//        -
//Methodes appelees:
//        -
//Commentaires:Cette méthode est appelée par les méthodes de DicoClasses
//qui ajoute une classe au DicoClasses. On s'est donné comme principe
//de créer une association (nomDeClasse,PaireListesIndicesVide) dans
//DicoEntree. Cette méthode est appelée par
//
// TDicoClasses::read,
//
//TDicoClasses::AjouterNouvelleEntree et par
//
//BOOL TDicoClasses::EstMembre(LPSTR,TSchemaClasse* &,BOOL &,HWND)

void TConteneurLiens::AjouterAssocDsDicoEntree(LPSTR szNomEntree)
{
        String    *str =  new String(szNomEntree);
        //créé paire listes vides
        TDeuxListsOrigExtr *listsDE =
                        new TDeuxListsOrigExtr() ;
        //crée l'assoc (nomClasse,paire liste vide)
        TNomEtListeDoubleAssoc *entreeDE =
                        new TNomEtListeDoubleAssoc(*str,*listsDE) ;
        //ajouter l'assoc dans DicoEntree
        pdicoEntree->add(*entreeDE) ;
}

//-------------
//<-        NOM DE LA METHODE:
// TConteneurLiens::AjouterLienHeritage méthode SURCHARGEE version I
//
TLienHeritageConceptuel * TConteneurLiens::
                AjouterLienHeritage(LPSTR szClasseDerivee,
                                                        LPSTR szClasseDeBase,HWND
hwnd)
{//début fonction

TSchemaClasse * pClasseOrigine ; //classe dérivée pour le moment
```

```
BOOL bOrigineCreated ;//requis pour EstMemmbre; pas nécessaire ici

TSchemaClasse * pClasseExtremite; //classe de base pour le moment

BOOL bExtremiteCreated ;//requis pour EstMemmbre ;pas nécessaire ici

int resOrigine =DicoClasses->
        EstMembre(szClasseDerivee,pClasseOrigine,bOrigineCreated,hwnd);

int resExtremite =DicoClasses->
        EstMembre(szClasseDeBase,pClasseExtremite,bExtremiteCreated,hwnd);
int resOriExt= resOrigine && resExtremite;
if( !resOriExt)
  {//début if
  //cette alternative du if est peu probable si la présente méthode
  //n'est appelée qu'à partir TFenSchemaClasse:NewBaseClAddHerit
  //TFenSchemaClasse::OldBaseClAddHerit comme c'est le cas pour le
  //moment
    char bufinfo[200]; strcpy(bufinfo,"l'une deux ou les deux classes ");
        strcat(bufinfo,szClasseDerivee) ;strcat(bufinfo," et ") ;
        strcat(bufinfo,szClasseDeBase) ;
        strcat(bufinfo,
                    " n'appartiennent pas au dictionnaire de classes ");
        MessageBeep(MB_ICONINFORMATION);
        MessageBox(hwnd,bufinfo,"Message O Etoile",

       MB_OK|MB_ICONINFORMATION) ;
        return NULL;
  }//fin if

 TLienHeritageConceptuel * pLienHerit =
  new TLienHeritageConceptuel (pClasseOrigine,pClasseExtremite) ;

int indTaleauLien = MettreLienDsTableauLiens(pLienHerit,hwnd);


if(indTaleauLien>=0)
 {//début if
  MajListesOriginesEtExtremites
                (
                                                    pClasseOrigine->NomClasse,
                                                    pClasseExtremite->NomClasse,
                    indTaleauLien,
                    hwnd
                                            );

  //retourne un pointeur vers le lien qui vient d'être céee
  //sert quand cette création  a été par déroulement du menu
  // Ajouter/LienHéritage de TFenSchemaClasse pour pouvoir donner une
  //valeur par défaut  aux points origine et extrémité du lien graphique
  //correspondant
  return  pLienHerit ;
 }//fin if(indTaleauLien>=0)

 return NULL ;
}//fin fonction

//-------------
//<-        NOM DE LA METHODE:
// TConteneurLiens::AjouterLienHeritage méthode SURCHARGEE version II
```

```
//
void TConteneurLiens::AjouterLienHeritage
      (LPSTR szClasseDerivee, LPSTR szClasseDeBase,
                  POINT  xyOriginGraphicLink, POINT  xyExtremeGraphicLink,
                  TPosLienGrSurIcone nPosLienGrSource,
                  TPosLienGrSurIcone nPosLienGrExtreme,
                                                                        HWND
hwndAjoutLien )

{//début fonction
 TSchemaClasse * pClasseOrigine ; //classe dérivée pour le moment

 BOOL bOrigineCreated ;//requis pour EstMemmbre; pas nécessaire ici

 TSchemaClasse * pClasseExtremite; //classe de base pour le moment

 BOOL bExtremiteCreated ;//requis pour EstMemmbre ;pas nécessaire ici

 int resOrigine =DicoClasses->
          EstMembre(szClasseDerivee,pClasseOrigine,bOrigineCreated,hwndAjoutLien);

 int resExtremite =DicoClasses->
          EstMembre(szClasseDeBase,pClasseExtremite,bExtremiteCreated,hwndAjoutLien),
int resOriExt= resOrigine && resExtremite;
if( !resOriExt)
   {//début if
   //cette alternative du if est peu probable si la présente méthode
   //n'est appelée qu'à partir TFenSchemaClasse:NewBaseClAddHerit
   //TFenSchemaClasse::OldBaseClAddHerit comme c'est le cas pour le
   //moment
    char bufinfo[200]; strcpy(bufinfo,"l'une deux ou les deux classes ");
          strcat(bufinfo,szClasseDerivee) ;strcat(bufinfo," et ") ;
          strcat(bufinfo,szClasseDeBase) ;
          strcat(bufinfo,
                        " n'appartiennent pas au dictionnaire de classes ");
          MessageBeep(MB_ICONINFORMATION);
          MessageBox(hwndAjoutLien,bufinfo,"Message O Etoile",

         MB_OK|MB_ICONINFORMATION) ;
    return ;
   }//fin if

 TLienHeritageConceptuel * pLienHerit =
  new TLienHeritageConceptuel (pClasseOrigine,pClasseExtremite) ;

int indTaleauLien = MettreLienDsTableauLiens(pLienHerit,hwndAjoutLien);
//négatif signifie le lien existait déjà dans le conteneur de liens
//donc l'enregistrement du lien n'a pu se faire
if(indTaleauLien>=0)
  {//début if
  MajListesOriginesEtExtremites
                  (
                                                pClasseOrigine->NomClasse,
                                                pClasseExtremite->NomClasse,
                        indTaleauLien,
                                                hwndAjoutLien
                                          );
  pLienHerit->EtablirPointsLienGraphique(xyOriginGraphicLink,

xyExtremeGraphicLink,
```

```
                                                                    nPosLienGrSource,
                                                                    nPosLienGrExtreme
                                                                    ) ;
  }//fin if(indTaleauLien>=0)


}//fin fonction

//-------------
//<-        NOM DE LA METHODE:
// TConteneurLiens::AjouterLienConceptuel
//
//
void TConteneurLiens::AjouterLienConceptuel(
                TLienConceptuel * pConceptualLink,
                POINT  xyOriginGraphicLink, POINT  xyExtremeGraphicLink,
                TPosLienGrSurIcone nPosLienGrSource,
                TPosLienGrSurIcone nPosLienGrExtreme,
                                                                    HWND
hwndAjoutLien )
{//début fonction
int indTaleauLien = MettreLienDsTableauLiens(pConceptualLink,
                                                          hwndAjoutLien);
 //négatif signifie le lien existait déjà dans le conteneur de liens
 //donc l'enregistrement du lien n'a pu se faire
 if(indTaleauLien>=0)
  {//début if
  MajListesOriginesEtExtremites
            (
                                        (pConceptualLink->OrigineLien() )->NomClasse,
                                        (pConceptualLink->ExtremiteLien() )->NomClasse,
                    indTaleauLien,
                                        hwndAjoutLien
                                    );
  pConceptualLink->EtablirPointsLienGraphique(xyOriginGraphicLink,

xyExtremeGraphicLink,
                                                                    nPosLienGrSource,
                                                                    nPosLienGrExtreme
                                                                    ) ;
  }//fin if(indTaleauLien>=0)


}//fin fonction


//-------------
//<-        NOM DE LA METHODE:
// TConteneurLiens::DisplayAllLinks

void TConteneurLiens::DisplayAllLinks(HDC hDCAllLinks)
{//début fonction
RContainerIterator allLinksIterator = ptableauliens->initIterator();

 while ( int(allLinksIterator) != 0 )
 {
        RObject allLinksAnObject = allLinksIterator++;
        if ( allLinksAnObject != NOOBJECT )
          ((RTLienConceptuel)(allLinksAnObject)).
```

```
                                                DessineToiDsGrapheSt (hDCAllLinks);
 }
 delete &allLinksIterator;


}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::AdjustGrLinksOnClassesRenamed
        -
void TConteneurLiens::AdjustGrLinksOnClassesRenamed(
                                HDC hDCLiensDeRenamed)
{//début fonction
 RContainerIterator listeIteratorRenamed
                            = pClassesJustBeenRenamed->initIterator();
 while ( int(listeIteratorRenamed) != 0 )
 {//début while
 TSchemaClasse & rschemaclassRenamed =
                            (TSchemaClasse &)listeIteratorRenamed++;
        if (rschemaclassRenamed!=NOOBJECT)
        {//début if
          AdjustOneGrLinkOnRenamedClass(
                                        &rschemaclassRenamed,hDCLiensDeRenamed) ;
        }//fin if(rschemaclassRenamed!=NOOBJECT)
 }//fin while
 delete &listeIteratorRenamed;
 //une fois établie les flèches repositionnées ,
 //on vide la liste des classes qui avaient fait l'objet d'un
 //renommage  sans bien sûr les détruire
 pClassesJustBeenRenamed->ownsElements(0);
 pClassesJustBeenRenamed->flush();

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::AdjustOneGrLinkOnRenamedClass

void TConteneurLiens::AdjustOneGrLinkOnRenamedClass
                                                (TSchemaClasse * pSchRenamed, HDC
hDCtoRename)
{//début fonction

TListeDeLiens*  listeLiensRenamed = ConteneurLiensHERIT->
        LiensDtClasseEstOrigine(pSchRenamed->NomClasse,hDCtoRename);
 RContainerIterator listRenamedIterator =
                        listeLiensRenamed ->initIterator();
 while ( int(listRenamedIterator) != 0 )
 {//début while
 TLienConceptuel & rlienconRenamed =
                                (TLienConceptuel &) listRenamedIterator++;
        if ( rlienconRenamed != NOOBJECT )
        {//début if
        rlienconRenamed.AdjustOrigOfGrLinkOnRenamedCl(pSchRenamed,

hDCtoRename);
        }//fin if ( rlienconRenamed != NOOBJECT )
 }//fin while
 delete &listRenamedIterator;

//////////////////
```

```
 TListeDeLiens* listeHeritRenamed = ConteneurLiensHERIT->
          LiensDtClasseEstExtremite(pSchRenamed->NomClasse,hDCtoRename);
RContainerIterator listeIteratorRenamed =listeHeritRenamed->
                                        initIterator();
while ( int(listeIteratorRenamed) != 0 )
 {//début while
 TLienConceptuel & rlienconRenamedExtr =
                              (TLienConceptuel &) listeIteratorRenamed++;
        if ( rlienconRenamedExtr != NOOBJECT )
        {//début if
        rlienconRenamedExtr.AdjustExtrOfGrLinkOnRenamedCl(pSchRenamed,

hDCtoRename);
        }//fin if
 }//fin while
delete &listeIteratorRenamed;


TListeDeLiens* listeLiensComposRenamed = ConteneurLiensCOMPOS->
          LiensDtClasseEstOrigine(pSchRenamed->NomClasse,hDCtoRename);
 RContainerIterator listComposRenamedIteratorOR =
                        listeLiensComposRenamed ->initIterator();
 while ( int(listComposRenamedIteratorOR) != 0 )
 {//début while
 TLienConceptuel & rlienconRenamed =
                                    (TLienConceptuel &) listComposRenamedIteratorOR++;
        if ( rlienconRenamed != NOOBJECT )
        {//début if
        rlienconRenamed.AdjustOrigOfGrLinkOnRenamedCl(pSchRenamed,

hDCtoRename);
        }//fin if ( rlienconRenamed!= NOOBJECT )
 }//fin while
 delete &listComposRenamedIteratorOR;

 TListeDeLiens* listeRenamed =ConteneurLiensCOMPOS->
          LiensDtClasseEstExtremite(pSchRenamed->NomClasse,hDCtoRename);
RContainerIterator listeComposIteratorRenamedEXT =listeRenamed->
                                        initIterator();
while ( int(listeComposIteratorRenamedEXT) != 0 )
 {//début while
 TLienConceptuel & rlienconRenamedExtr =
                              (TLienConceptuel &) listeComposIteratorRenamedEXT++;
        if ( rlienconRenamedExtr != NOOBJECT )
        {//début if
        rlienconRenamedExtr.AdjustExtrOfGrLinkOnRenamedCl(pSchRenamed,

hDCtoRename);
        }//fin if
 }//fin while
delete &listeComposIteratorRenamedEXT;


TListeDeLiens* listeLiensReferRenamed = ConteneurLiensREFER->
          LiensDtClasseEstOrigine(pSchRenamed->NomClasse,hDCtoRename);
 RContainerIterator listReferRenamedIteratorOR =
                        listeLiensReferRenamed ->initIterator();
 while ( int(listReferRenamedIteratorOR) != 0 )
 {//début while
```

```
 TLienConceptuel & rlienconRenamed =
                                    (TLienConceptuel &) listReferRenamedIteratorOR++;
        if ( rlienconRenamed != NOOBJECT )
        {//début if
        rlienconRenamed.AdjustOrigOfGrLinkOnRenamedCl(pSchRenamed,

hDCtoRename);
        }//fin if ( rlienconRenamed!= NOOBJECT )
 }//fin while
 delete &listReferRenamedIteratorOR;

 TListeDeLiens*  listeReferRenamed =ConteneurLiensREFER->
          LiensDtClasseEstExtremite(pSchRenamed->NomClasse,hDCtoRename);
RContainerIterator listeReferIteratorRenamedEXT =listeReferRenamed->
                                        initIterator();
while ( int(listeReferIteratorRenamedEXT) != 0 )
 {//début while
 TLienConceptuel & rlienconRenamedExtr =
                                (TLienConceptuel &) listeReferIteratorRenamedEXT++;
        if ( rlienconRenamedExtr != NOOBJECT )
        {//début if
        rlienconRenamedExtr.AdjustExtrOfGrLinkOnRenamedCl(pSchRenamed,

hDCtoRename);
        }//fin if
 }//fin while
delete &listeReferIteratorRenamedEXT;


}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::DisplayNewLinksFromTFenSch
//
//
void TConteneurLiens::SetDefaultsGrLinksDeTFenSch(HDC hDCLiensDeTFenSCh)
{//début fonction
 RContainerIterator listeIteratorTFenSch
        =pLiensHeritDeTFenSchemaClasse->initIterator();
 while ( int(listeIteratorTFenSch) != 0 )
 {//début while
 TLienConceptuel & rlienconTFenSch =
                                (TLienConceptuel &)listeIteratorTFenSch++;
        if ( rlienconTFenSch!= NOOBJECT )
        {//début if

                SetDefautsGrForOneHLink( &rlienconTFenSch,hDCLiensDeTFenSCh)
        }//fin if
 }//fin while
 delete &listeIteratorTFenSch;
 //une fois établie les flèches par défaut ,on vide la liste des liens
 //sans bien sûr les détruire
 pLiensHeritDeTFenSchemaClasse->ownsElements(0);
 pLiensHeritDeTFenSchemaClasse->flush();
}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::SetDefautsGrForOneHLink
//
void TConteneurLiens::
```

```
                    SetDefautsGrForOneHLink(TLienConceptuel * pLienTFenSch,HDC)
{//début fonction

//on établit ici une flèche arbitraire entre une icône de classe source
 // et une icône de classe extrémité; c'est une action par défaut faute
 //la TFenGrapheStatique sous les yeux; on pourra éventuellement si ce
 // lien ne convient pas le redessiner manuellement dans
 // la TFenGrapheStatique


 POINT ptOrig,ptExtrem ;
 RECT  rectOrig ,rectExtrem ;
 rectOrig =pLienTFenSch->OrigineLien()->FenSchemaClasse->
                    rIconeClasse.Forme;
 ptOrig.x = rectOrig.right;
 ptOrig.y = rectOrig.top ;
 rectExtrem =pLienTFenSch->ExtremiteLien()->FenSchemaClasse->
                     rIconeClasse.Forme;
 ptExtrem.x = rectExtrem.right;
 ptExtrem.y = rectExtrem.bottom ;
 pLienTFenSch->EtablirPointsLienGraphique (ptOrig,ptExtrem,0,0 ) ;

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::AddClassJustRenamedInList
//
//Commentaires:
//on demande au conteneur de liens de maintenir une liste de classe
//qui viennent d'être renommées pour ensuite lors de l'affichage
//de la zone client de TFenGrapheStatique REAJUSTER les points
//origines et extrémités des liens aboutissant sur la classe
//qui est l'objet du renommage (cela permet des liens qui pendent)
void TConteneurLiens::AddClassJustRenamedInList( TSchemaClasse * pSchToAdd)
{//début fonction

pClassesJustBeenRenamed->add( *pSchToAdd ) ;

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::AjouterLienHdeTFenSCh

void TConteneurLiens::
          AjouterLienHdeTFenSCh(TLienConceptuel * pLienTFenCh)
{//début fonction
if (pLienTFenCh )
  pLiensHeritDeTFenSchemaClasse->add(*pLienTFenCh) ;

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::MettreLienDsTableauLiens(TLienConceptuel *,HWND)

int TConteneurLiens::MettreLienDsTableauLiens
                                        (TLienConceptuel *plinkhole,HWND hwndhole)
{//début fonction

TFenPrinc *pFenPrinc =
          (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
```

```
int indice ;

if(!(ptableauliens->hasMember(*plinkhole)) )
  {//début if
        pFenPrinc->IsDirty = TRUE ;
        ptableauliens->Ajouter (*plinkhole,indice) ;
   return indice ;
  }//fin if
  else
  {//début else
   char bufident[200] ;

   strcpy(bufident,"le lien statique '") ;
   strcat(bufident,plinkhole->OrigineLien()->NomClasse);
   strcat(bufident, " hérite de ");
   strcat(bufident,plinkhole->ExtremiteLien()->NomClasse);
        strcat(bufident,"'\n est déjà présent dans le conteneur de liens") ;
   for(int i = 0; i<=3;i++) MessageBeep(MB_ICONHAND) ;
        MessageBox(hwndhole,bufident," O Etoile ",MB_ICONSTOP|MB_OK) ;

        return -1 ;
  }//fin else


}//fin fonction
//------------
//<-        NOM DE LA METHODE:
//TConteneurLiens::MajListesOriginesEtExtremites
        -
//Commentaires:méthode surchargée
//

void TConteneurLiens::MajListesOriginesEtExtremites(LPSTR szOrigine,

LPSTR szExtremite, int nIndArray,  HWND hwndmaj)
{//début fonction

 pdicoEntree->MajListesOriginesEtExtremites
                                                (
                                                        szOngine,
                                                        szExtremite,
                                                        nIndArray,
                                                        hwndmaj
                                                );


}//fin fonction

//<-        NOM DE LA METHODE
//TConteneurLiens::SupprimerLienConceptuel(TLienConceptuel*,HWND)

void TConteneurLiens::SupprimerLienConceptuel(TLienConceptuel * plien,HWND hwd)
{//début fonction

//Recherche de l'indice (entrée de l'objet *plien ) dans le tableau
//avant de le détacher du tableau
int slotNumber = ptableauliens->TrouverIndiceDsTableau(*plien) ;
//je détache le lien du tableau de lien sans le détruire lui-même
ptableauliens->ownsElements(0);
ptableauliens->detach(*plien);
```

```
//Rechercher la liste des indices (origines) du tableau de liens pour la
//classe origine du lien : cette liste perd un élément
//Convention : A hérite de B
//A est origine du lien  et B est extrémité du lien
TListeIndices  * plistOrig =
                  pdicoEntree-> ObtenirListeIndicesOrigines(
                                    (plien->OrigineLien())->NomClasse,
                                    hwd ) ;
//je détache l'indice de cette liste des indices origines
if(plistOrig != NULL)
 {//début if
        plistOrig->ownsElements(0);
        plistOrig->detach(* new TEntier(slotNumber));
 }//fin if
 else
 {//début else
  MessageBeep(MB_ICONEXCLAMATION);
  MessageBox(hwd,"ObtenirListeIndicesOrigines"
  " a retourné un pointeur NULL","O Etoile - SupprimerLien",
                                          MB_ICONEXCLAMATION|MB_OK);
 }//fin else



//Rechercher la liste des indices (extrémités) du tableau de liens pour la
//classe extrémité du lien : cette liste perd un élément
//Convention : A hérite de B
//A est origine du lien  et B est extrémité du lien
TListeIndices  * plistExtr=
                  pdicoEntree-> ObtenirListeIndicesExtremites(
                                    (plien->ExtremiteLien() )->NomClasse,
                                    hwd ) ;
//je détache l'indice de cette liste des indices extrémités
if(plistExtr != NULL)
 {//début if
        plistExtr->ownsElements(0);
        plistExtr->detach(* new TEntier(slotNumber));
 }//fin if
 else
 {//début else
  MessageBeep(MB_ICONEXCLAMATION);
  MessageBox(hwd,"ObtenirListeIndicesExtremites"
  " a retourné un pointeur NULL","O Etoile - SupprimerLien",
                                          MB_ICONEXCLAMATION|MB_OK);
 }//fin else


}//fin fonction

//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>        NOM DE LA METHODE:TConteneurLiens:: build
PTStreamable TConteneurLiens:: build()
 {//début fonction

 return new TConteneurLiens(streamableInit) ;
```

```
}//fin fonction

//>>        NOM DE LA METHODE:
//                TConteneurLiens  write(Ropstream )
//
void TConteneurLiens::write(Ropstream os)
{//début fonction

pdicoEntree->write(os) ;
ptableauliens->write(os) ;

}//fin fonction

//>>        NOM DE LA METHODE:
//                TConteneurLiens .read(Ripstream )
//
Pvoid TConteneurLiens::read(Ripstream is)
{//début fonction

 pdicoEntree->read(is) ;
 ptableauliens->read(is) ;

 return this ;
}//fin fonction



//************************************************************************
//************************************************************************
//              METHODES DE LA CLASSE:TDicoEntree
//************************************************************************

//++++++++-------- CONSTRUCTEURS ET DESTRUCTEUR
//-------------
//>>        NOM DE LA METHODE
//  TDicoEntree::TDicoEntree(StreamableInit)
//
TDicoEntree::TDicoEntree(StreamableInit)
{
}
 //-------------
//>>        NOM DE LA METHODE TDicoEntree::streamableName()
//
//
const Pchar TDicoEntree::streamableName() const
{//début fonction

return "TDicoEntree" ;

}//fin fonction

//                              -+-+-+-+-+-+-+-+-+-PROTEGE-+-+-+-+-+-+

//                              ++++++++++++++++++PUBLIQUE+++++++++++++++


//[]-> Methodes d'information
//-------------
//->        NOM DE LA METHODE:ObtenirListeIndicesOrigines
//
```

```
TListeIndices  * TDicoEntree::
          ObtenirListeIndicesOrigines(LPSTR szClassName,HWND hfen)
{//début fonction

String  * pstrIndOrig = new String( szClassName );
TNomEtListeDoubleAssoc & rAssocIndOrig=
                  (TNomEtListeDoubleAssoc &) lookup(*pstrIndOrig) ;

if( rAssocIndOrig== NOOBJECT )
                  {
                   char buf[180] ; strcpy(buf,szClassName);
                   strcat(buf,
                   "  non trouvée dans le dictionnaire DicoEntree ");
                   MessageBeep(MB_ICONEXCLAMATION);
                   MessageBox(hfen,buf,
                    "O Etoile Message -ObtenirListeIndicesOrigines ",
                             MB_ICONEXCLAMATION|MB_OK) ;

                   return NULL ;
                  }
 else
  {
  TDeuxListsOrigExtr*   pdblelistIndOrig =
                        (TDeuxListsOrigExtr*)  & (rAssocIndOrig.value())  ;

  TListeIndices  * plisteIndOrig = pdblelistIndOrig->listeOrigines() ;

          return  plisteIndOrig ;
  }

}//fin fonction
//-------------
//->        NOM DE LA METHODE:ObtenirListeIndicesExtremites

TListeIndices  * TDicoEntree::
          ObtenirListeIndicesExtremites(LPSTR szUnNomClasse,HWND hfenetre)
{//début fonction

 String  * pstrIndExtr = new String( szUnNomClasse);
TNomEtListeDoubleAssoc & rAssocIndExtr=
                  (TNomEtListeDoubleAssoc &) lookup(*pstrIndExtr) ;

if( rAssocIndExtr== NOOBJECT )
                  {
                   char buf[180] ; strcpy(buf,szUnNomClasse);
                   strcat(buf,
                   "  non trouvée dans le dictionnaire DicoEntree ").
                   MessageBeep(MB_ICONEXCLAMATION);
                   MessageBox(hfenetre,buf,
                    "O Etoile Message - ObtenirListeIndicesExtremites",
                             MB_ICONEXCLAMATION|MB_OK) ;
      return NULL ;
                  }
 else
  {
  TDeuxListsOrigExtr*   pdblelistIndExtr =
                        (TDeuxListsOrigExtr*)  & (rAssocIndExtr.value())  .

  TListeIndices  * plisteIndExtr = pdblelistIndExtr->listeExtremites();
  return  plisteIndExtr ;
```

```
  }

}//fin fonction


//[]<- Methodes de manipulation
//-------------
//<-        NOM DE LA METHODE:TDicoEntree::RenommerClasse
//
void TDicoEntree::RenommerClasse(LPSTR AncNomDE,LPSTR NouvNomDE)
{//début fonction
//je récupère l'adresse de paire de listes d'indices du tableau
//à trous (liste des origines et liste des extrémités) dans
//pdoubleliste et je détache l'ancienne assoc du dico d'entrée
TDeuxListsOrigExtr  * pdoubleliste ;
DetacherClasse(AncNomDE,pdoubleliste);

//je recrée une autre String avec le nouveau nom que j'associe avec
//l'objet paire de listes (TDeuxListsOrigExtr) dont je viens
//d'obtenir l'adresse et je range l'association dans Le dico d'entrée
String * pstrdico = new String( NouvNomDE);
TNomEtListeDoubleAssoc *entreeDE =
 new TNomEtListeDoubleAssoc( *pstrdico, *pdoubleliste);
          add(*entreeDE);


}//fin fonction




//-------------
//        NOM DE LA METHODE: TDicoEntree::DetacherClasse
//
void TDicoEntree::DetacherClasse(LPSTR NomCIDE,
                                          TDeuxListsOrigExtr * & pdblelistDE)
{

String  * pstrDE = new String (NomCIDE) ;
TNomEtListeDoubleAssoc & rAssocDE =
                  (TNomEtListeDoubleAssoc &) lookup(*pstrDE) ;
        if( rAssocDE== NOOBJECT )
                  {
                   char bufdetach[180] ; strcpy(bufdetach,NomCIDE);
                   strcat(bufdetach,
                   "  non trouvée dans le dictionnaire DicoEntree ");
                   MessageBeep(MB_ICONEXCLAMATION);
                   MessageBox(GetApplicationObject()->MainWindow->HWindow,
                     bufdetach,"O Etoile Message ",MB_ICONEXCLAMATION|MB_OK) ;
                  }
 else
  {
  //récupère l'adresse de l'assoc pour la renvoyer à l'appelant
  pdblelistDE= (TDeuxListsOrigExtr*)  & (rAssocDE.value())  ;

  //enlève l'assoc du conteneur mais ne la détruit pas !!
  ownsElements(0);
  detach(rAssocDE ) ;

  }
```

```
}

//-------------
//          NOM DE LA METHODE:
//   TDicoEntree::MajListesOriginesEtExtremites
//
void TDicoEntree::MajListesOriginesEtExtremites(LPSTR szOrig, LPSTR szExtr,
                                                                                    int
                                                nIdArray,

                                                HWND hwdmaj)
{//début fonction


String  * pstrOrig = new String (szOrig) ;
TNomEtListeDoubleAssoc& rAssocOrig =
                    (TNomEtListeDoubleAssoc&) lookup(*pstrOrig) ;

String  * pstrExtr= new String (szExtr) ;
TNomEtListeDoubleAssoc& rAssocExtr=
                    (TNomEtListeDoubleAssoc&) lookup(*pstrExtr) ;

int resOrigExtr = (rAssocOrig== NOOBJECT )||(rAssocExtr == NOOBJECT ) ;

if(!resOrigExtr )
  {//débuf if
  ((TDeuxListsOrigExtr &) (rAssocOrig.value()) ).
          MajListeOrigine(szOrig,nIdArray,hwdmaj );

   ((TDeuxListsOrigExtr &) (rAssocExtr.value()) ).
   MajListeExtremite(szExtr,nIdArray,hwdmaj );

  }//fin if
  else
   {//début else

   //alternative peu probable étant donné que l'on s'est fixé
   //comme discipline de créer une paire de liste vide dans
   //DicoEntree à chaque fois que l'on crée une nouvelle classe et
   //l'on l'ajoute à DicoClasses

   char buforex[250] ; strcpy(buforex,"l'une des classes ");
   strcat(buforex,szOrig);strcat(buforex," ou ");strcat(buforex,szExtr);
   strcat(buforex,"ne possède pas valeur (value) dans DicoEntree");
   MessageBeep(MB_ICONEXCLAMATION);
   MessageBox(hwdmaj,buforex,"O Etoile Message ",MB_ICONEXCLAMATION|MB_OK) ;

   }//fin else

}//fin fonction


//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>          NOM DE LA METHODE:TDicoEntree:: build
//
PTStreamable TDicoEntree:: build()
 {//début fonction

 return new TDicoEntree(streamableInit) ;
```

```
}//fin fonction

//>>        NOM DE LA METHODE:
//                TDicoEntree:: write(Ropstream )
//

void TDicoEntree::write(Ropstream os)
{//début fonction

 RContainerIterator writeIterator = initIterator();

  // le dico d'entrée est reconstitué en même temps que le dico
  // de classes est en train d'être lue avec une paire de liste
  // indices origines et extrémités vides; donc inutile de sauver
  // getItemsInContainer puiqu'il est mis à jour à chaque ajout
  // d'une assoc par TDicoClasses  read
  // os << getItemsInContainer();

  while ( int(writeIterator) != 0 )
  {
        RObject writeObject = writeIterator++;
        if ( writeObject != NOOBJECT )
          ((PTNomEtListeDoubleAssoc  )(&writeObject))->write (os);
  }
  delete &writeIterator;

}//fin fonction

//>>        NOM DE LA METHODE:
//                TDicoEntree::read(Ripstream )
//
//


Pvoid TDicoEntree::read(Ripstream is)
{//début fonction
RContainerIterator readIterator = initIterator();

        while ( int(readIterator) != 0 )
        {//début while
                RObject readObject = readIterator++;
                if ( readObject != NOOBJECT )
                        ((PTNomEtListeDoubleAssoc    )(&readObject))->read (is);
        }//fin while

delete &readIterator;
return this ;
}//fin fonction


//*****************************************************************
//*****************************************************************
//              METHODES DE LA CLASSE: TNomEtListeDoubleAssoc
//*****************************************************************

//                                  ----------PRIVE----------
//<< >> << >> << >>Methode de gestion des flux
//----------
//>>        NOM DE LA METHODE
```

```
//     TNomEtListeDoubleAssoc::TNomEtListeDoubleAssoc(StreamableInit)

TNomEtListeDoubleAssoc::TNomEtListeDoubleAssoc(StreamableInit)
:Association(NOOBJECT,NOOBJECT)
 {//début fonction


 }//fin fonction


//>>          NOM DE LA METHODE:TNomEtListeDoubleAssoc::streamableName()


const Pchar TNomEtListeDoubleAssoc::streamableName() const
{//début fonction

return "TNomEtListeDoubleAssoc" ;

}//fin fonction



//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>          NOM DE LA METHODE:TNomEtListeDoubleAssoc:: build

PTStreamable TNomEtListeDoubleAssoc:: build()
 {//début fonction

 return new TNomEtListeDoubleAssoc(streamableInit) ;

 }//fin fonction

//>>          NOM DE LA METHODE:
//                TNomEtListeDoubleAssoc:: write(Ropstream )
//

void TNomEtListeDoubleAssoc::write(Ropstream os)
{//début fonction

 ((TDeuxListsOrigExtr & ) value())   write ( os );

}//fin fonction

//>>          NOM DE LA METHODE:
//                TNomEtListeDoubleAssoc::read(Ripstream )
Pvoid TNomEtListeDoubleAssoc::read(Ripstream is)
{//début fonction
        ((TDeuxListsOrigExtr  & ) value()) . read ( is ) ;

return this ;
}//fin fonction


//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE: TDeuxListsOrigExtr
//*************************************************************************

//                                  ----------PRIVE----------
```

```
//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>          NOM DE LA METHODE:
//   TDeuxListsOrigExtr::TDeuxListsOrigExtr(StreamableInit)
//
TDeuxListsOrigExtr::TDeuxListsOrigExtr(StreamableInit)
 {//début fonction
 plisteOrigines = new TListeIndices();
 plisteExtremites = new TListeIndices();

 }//fin fonction


//>>          NOM DE LA METHODE:TDeuxListsOrigExtr::streamableName()

const Pchar TDeuxListsOrigExtr::streamableName() const
{//début fonction

return "TDeuxListsOrigExtr" ;

}//fin fonction

//                                  ++++++++++++++++++PUBLIQUE+++++++++++++++
//-------------
//<-          NOM DE LA METHODE:TDeuxListsOrigExtr::MajListeOrigine
//
//
void TDeuxListsOrigExtr::MajListeOrigine(LPSTR szOrig,int nIdArrayor,
                                                                        HWND )
{//début fonction
TEntier *pnOrig= new TEntier(nIdArrayor);
plisteOrigines->add(*pnOrig);
//DEBUT CODE DE DEBUGGAGE ET DE TEST
 RContainerIterator listorIterator =plisteOrigines-> initIterator();
 char bufentor[150] ;
 char bufentor1[130] ;   bufentor1[0]=0;
 char bufenttempor1[LGMAXNOM] ;
 strcpy(bufentor," voici les indices ORIGINE pour la classe ");
 strcat(bufentor,szOrig);
  while ( int(listorIterator) != 0 )
  {//début while
  TEntier & listorEntier= (TEntier &) listorIterator++;
          if ( listorEntier != NOOBJECT )
          {//début if
           int nor ;
           nor= listorEntier ;
           wsprintf(bufenttempor1,"%d",nor);
           strcat(bufentor1,bufenttempor1);
           strcat(bufentor1,"  ");
          }//fin if
  }//fin while
// MessageBox(hwdmajor,bufentor1,bufentor,MB_OK) ;
 delete &listorIterator;
 //FIN CODE DE DEBUGGAGE ET DE TEST


}//fin fonction


//-------------
```

```
//<-        NOM DE LA METHODE:TDeuxListsOrigExtr::MajListeExtremite
//

void TDeuxListsOrigExtr::MajListeExtremite(LPSTR szExtr,int nIdArrayex,

HWND )
{//début fonction
TEntier *pnExtr = new TEntier(nIdArrayex);
plisteExtremites->add(*pnExtr);

//DEBUT CODE DE DEBUGGAGE ET DE TEST
 RContainerIterator listexIterator =plisteExtremites-> initIterator();
 char bufentex[150] ;
 char bufentex1[130] ;    bufentex1[0] = 0;
 char bufentextemp1[LGMAXNOM] ;
 strcpy(bufentex," voici les indices EXTREMITES pour la classe ");
 strcat(bufentex,szExtr);
  while ( int(listexIterator) != 0 )
  {//début while
  TEntier & listexEntier= (TEntier &) listexIterator++;
          if ( listexEntier != NOOBJECT )
          {//début if
           int nex ;
           nex = listexEntier ;
           wsprintf(bufentextemp1,"%d",nex);
           strcat(bufentex1,bufentextemp1);
           strcat(bufentex1,"  ");
          }//fin if
  }//fin while
// MessageBox(hwdmajex,bufentex1,bufentex,MB_OK) ;
 delete &listexIterator;
 //FIN CODE DE DEBUGGAGE ET DE TEST



}//fin fonction


//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>        NOM DE LA METHODE:TDeuxListsOrigExtr:: build
PTStreamable TDeuxListsOrigExtr:: build()
 {//début fonction

 return new TDeuxListsOrigExtr(streamableInit) ;

 }//fin fonction

//>>        NOM DE LA METHODE:
//              TDeuxListsOrigExtr:: write(Ropstream )
//
//
void TDeuxListsOrigExtr::write(Ropstream os)
{//début fonction
//sauver les indices ORIGINES
        RContainerIterator writeOrigIterator = plisteOrigines->initIterator();
        ecrire ("Nombre d'indices origine        :   %3d\r\n",
                        plisteOrigines->getItemsInContainer(),
        os );
```

```
        while ( int(writeOrigIterator) != 0 )
        {
        TEntier & rUnIndice= (TEntier &) writeOrigIterator++;
        if ( rUnIndice != NOOBJECT )
         {//début if
                ecrire ("Indice  origine                :   %3d\r\n",
                        rUnIndice,
                        os );
                // os  << (int ) rUnIndice ;
         }//fin if
        }
        delete &writeOrigIterator;

//sauver les indices EXTREMITES
        RContainerIterator writeExtrIterator = plisteExtremites->initIterator();
        ecrire ("Nombre d'indices extreme       :   %3d\r\n",
                        plisteExtremites->getItemsInContainer(),
        os );

        while ( int(writeExtrIterator) != 0 )
        {
        TEntier & rUnIndice= (TEntier &) writeExtrIterator++;
        if ( rUnIndice != NOOBJECT )
         {//début if
                ecrire ("Indice  extrémité              :   %3d\r\n",
                        rUnIndice,
                        os );

         //       os  << (int ) rUnIndice ;
    }//fin if
        }
        delete &writeExtrIterator;
}//fin fonction

//>>        NOM DE LA METHODE:
//              TDeuxListsOrigExtr::read(Ripstream )

Pvoid TDeuxListsOrigExtr::read(Ripstream is)
{//début fonction
countType       CardinalIndicesOrigines,
                        CardinalIndicesExtremes;
int             nUnIndice ;

int             nLongueur = strlen
                                ("Nombre d'indices origine      :   %3d\r\n") ;
char            * szTamponOr = new char [nLongueur + 1],
                        * szTamponEx = new char [nLongueur + 1] ;
                         //delete en fin de fonction
char            szNbreIndiceEtIndicesOr [4] ,
                        szNbreIndiceEtIndicesEx [4] ;
int             NbreCar = 3 . // correspond à %3d
int             PosInit = nLongueur - NbreCar - 2 ;

// lire les indices origines
        lire            (szTamponOr, nLongueur, is) ;
        extraire        (szTamponOr, PosInit, NbreCar) ;
        strcpy          (szNbreIndiceEtIndicesOr, szTamponOr) ;

        CardinalIndicesOrigines = atoi (szNbreIndiceEtIndicesOr) ;
```

```
   for ( int i = 0 ; i < CardinalIndicesOrigines ; ++i )
        {//début for
                // on réutilise les mêmes variables, qui cependant contiennent
      // d'autres informations ; les longueurs restent identiques.
                lire            (szTamponOr, nLongueur, is) ;
                extraire        (szTamponOr, PosInit, NbreCar) ;
                strcpy          (szNbreIndiceEtIndicesOr, szTamponOr) ;
                nUnIndice       = atoi(szNbreIndiceEtIndicesOr) ;
                // is >> nUnIndice ;
                TEntier *pnOrig = new TEntier(nUnIndice);
                plisteOrigines->add(*pnOrig);  //add met à jour itemsInContainer
        }//fin for

// lire les indices EXTREMES
// nLongueur n'a pas à être recalculé, c'es id. à indice Origine
        lire            (szTamponEx,nLongueur, is) ;
        extraire        (szTamponEx, PosInit, NbreCar) ;
        strcpy          (szNbreIndiceEtIndicesEx, szTamponEx) ;

        CardinalIndicesExtremes = atoi (szNbreIndiceEtIndicesEx) ;

        for (  i = 0 ; i < CardinalIndicesExtremes ; ++i )
        {//début for
                // on réutilise les mêmes variables. qui cependant contiennent
      // d'autres informations ; les longueurs restent identiques.
                lire            (szTamponOr, nLongueur, is) ;
                extraire        (szTamponOr, PosInit, NbreCar) ;
                strcpy          (szNbreIndiceEtIndicesOr, szTamponOr) ;
                nUnIndice       = atoi(szNbreIndiceEtIndicesOr) ;
                // is >> nUnIndice ;
                TEntier *pnExtr= new TEntier(nUnIndice);
                plisteExtremites->add(*pnExtr);//add met à jour itemsInContainer
        }//fin for
delete szTamponOr, szTamponEx ;
return this ;
}//fin fonction

//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE: TArrayWithHolesOfLinks
//*************************************************************************

//                                  ----------PRIVE----------
//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>        NOM DE LA METHODE
//  TArrayWithHolesOfLinks::TArrayWithHolesOfLinks(StreamableInit)
TArrayWithHolesOfLinks::TArrayWithHolesOfLinks(StreamableInit)
: Array( 10, 0, 10 )
 {//début fonction

 }//fin fonction


//>>        NOM DE LA METHODE TArrayWithHolesOfLinks::streamableName()
//
const Pchar TArrayWithHolesOfLinks::streamableName() const
{//début fonction

return "TArrayWithHolesOfLinks" ;
```

```
}//fin fonction

//                              ++++++++++++++++++PUBLIQUE+++++++++++++++

//[]<- Methodes de manipulation
//-------------
//<-        NOM DE LA METHODE:TArrayWithHolesOfLinks::Ajouter
//
//
void TArrayWithHolesOfLinks::Ajouter(Object & toAdd, int & index)
{

int IndiceTrouOuInserer =lowerbound ;
//lastElementIndex++;
        while( IndiceTrouOuInserer <= upperbound &&
                  ptrAt( IndiceTrouOuInserer ) != ZERO
      )
                IndiceTrouOuInserer++;

        if( IndiceTrouOuInserer > upperbound )
                reallocate( IndiceTrouOuInserer - lowerbound + 1 );

        setData( IndiceTrouOuInserer, &toAdd );
        index = IndiceTrouOuInserer ;
        if( IndiceTrouOuInserer == lastElementIndex + 1 )
                                        lastElementIndex = IndiceTrouOuInserer ;
  itemsInContainer++;
        CHECK( itemsInContainer > 0 );

}
//-------------
//<-        NOM DE LA METHODE:
//TArrayWithHolesOfLinks::AjouterDomaine

int TArrayWithHolesOfLinks::AjouterDomaine(LPSTR szNomDom ,LPSTR szExprDom)
{//début fonction
TDomaine * pUnDomaine = new TDomaine(szNomDom,szExprDom) ;
 if(!hasMember(*pUnDomaine) )
  {//début if (!hasMember(*pUnDomaine) )
    int Indice;
         Ajouter( *pUnDomaine , Indice) ;
         return Indice ;
  }//fin if (!hasMember(*pUnDomaine) )
  return -1;
}//fin fonction
//-------------
//<-        NOM DE LA METHODE
//TArrayWithHolesOfLinks::AjouterPropriete
//

int TArrayWithHolesOfLinks::AjouterPropriete(LPSTR szPropName,
                                                        LPSTR szDomName)
{//début fonction

TDomaine * pdom = new TDomaine(szDomName,"");
int resIndice =ConteneurDomaines->TrouverIndiceDsTableau(*pdom) ;
delete  pdom ;

TPropriete * pProp = new TPropriete(szPropName,resIndice);
```

```
 if(!hasMember(*pProp) )
  {//début if (!hasMember(*pProp) )
   int Indice;
         Ajouter( *pProp, Indice) ;
         return Indice ;
  }//fin if (!hasMember(*pProp) )
  return -1;

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:TArrayWithHolesOfLinks::detach(int,DeleteType)

void TArrayWithHolesOfLinks::detach( int atIndex, DeleteType dt )
{//début fonction

 char bufAtIndex[10] ; wsprintf(butAtIndex,"%d",atIndex) ;

        if( ptrAt(atIndex) != ZERO )
     {
                if( delObj(dt) )
                        delete ptrAt(atIndex);
                itemsInContainer--;
                }
  // removeEntry(atIndex); on supprime le tassement du tableau fait
  // par detach
        setData( atIndex, ZERO )    // à l'indice absolue atIndex
                                                        //il y a un trou
        if (atIndex==lastElementIndex ) lastElementIndex = atIndex - 1 ;

   CHECK( itemsInContainer != UINT_MAX );
}//fin fonction

//-------------
//<-        NOM DE LA METHODE:
// TArrayWithHolesOfLinks::detach( Object& ,DeleteType)

void TArrayWithHolesOfLinks::detach( Object& toDetach, DeleteType dt )
{
        detach( find( toDetach ), dt );
}
//-------------
//<-        NOM DE LA METHODE
// TArrayWithHolesOfLinks::AfficherNomsDomsDsBoite
        -
void  TArrayWithHolesOfLinks::AfficherNomsDomsDsBoite
                        (TDialog* pDial, int ID_BteListe)
{//début fonction
RContainerIterator affBteListeIterator = initIterator();

 while ( int(affBteListeIterator) != 0 )
 {//début while

        RTDomaine affBteListeAnObject =
                                                (RTDomaine)affBteListeIterator++;
        if ( affBteListeAnObject != NOOBJECT )
  {//début if

        pDial->SendDlgItemMsg(ID_BteListe,LB_ADDSTRING,NULL,
                        (DWORD) (affBteListeAnObject GetNomDomaine()) )
```

```
        }//fin if( affBteListeAnObject != NOOBJECT )
 }//fin while

 delete &affBteListeIterator;

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
// TArrayWithHolesOfLinks::AfficherNomsPropsDsBoiteListe

void TArrayWithHolesOfLinks::AfficherNomsPropsDsBoiteListe
                (TDialog* pDial, int ID_BteListe)
{//début fonction
RContainerIterator affBteListeIterator = initIterator();

 while ( int(affBteListeIterator) != 0 )
 {//début while

        RTPropriete affBteListeAnObject =
                                                (RTPropriete)affBteListeIterator++;
        if ( affBteListeAnObject != NOOBJECT )
  {//début if

        pDial->SendDlgItemMsg(ID_BteListe,LB_ADDSTRING,NULL,
                        (DWORD) (affBteListeAnObject.GetNomPropriete())  ) ;

        }//fin if( affBteListeAnObject != NOOBJECT )
 }//fin while

 delete &affBteListeIterator;


}//fin fonction
//

//<< >> << >> << >>Methode de gestion des flux
//-------------
//>>        NOM DE LA METHODE:TArrayWithHolesOfLinks:: build
PTStreamable TArrayWithHolesOfLinks:: build()

 {//début fonction

 return new TArrayWithHolesOfLinks(streamableInit) ;

 }//fin fonction

//>>        NOM DE LA METHODE:
//              TArrayWithHolesOfLinks:: write(Ropstream )
//
void TArrayWithHolesOfLinks::write(Ropstream os)
{//début fonction

// sauver le nombre d'entrée dans la tableau de liens
// les trous compris qui correspond à lastElementIndex+1
// car notre lowerbound commence à 0
        ecrire ("Nbre d'entrées Tabl. liens concept :    %3d\r\n",
                        lastElementIndex + 1,
                        os ) ;

 for(int IndiceLien = 0 ;
```

```
                        IndiceLien < (lastElementIndex + 1 );
                    IndiceLien++)
 {//début for
        RTLienConceptuel writeAnObject =
                                                (RTLienConceptuel) objectAt(IndiceLien) ;

        if ( writeAnObject != NOOBJECT )
        {//début if
        //un lien existe à cet endroit
  //sauver ce fait dans le fichier à des fins de relectures

        ecrireTexte ("Nature du lien conceptuel        :     L\r\n" os) ;

        //sauver le nom de la classe origine
        TSchemaClasse* pScOr = writeAnObject.OrigineLien();
        ecrire ("Longueur du nom de classe origine  :    %3d\r\n",
                        lstrlen(pScOr->NomClasse),
                        os ) ;

  char tmpNom [LGMAXNOM + 1] ;
        lstrcpy  (tmpNom,pScOr->NomClasse ) ;
        ecrireTexte (tmpNom, os ) ;
  ecrireTexte("\r\n", os) ; // pour faire beau

    //et sauver le nom de la classe extrémité
          TSchemaClasse* pScExt = writeAnObject.ExtremiteLien();
        ecrire ("Longueur du nom de classe extremite:    %3d\r\n",
                        lstrlen(pScExt->NomClasse),
                        os ) ;

        lstrcpy  (tmpNom,pScExt->NomClasse ) ;
        ecrireTexte (tmpNom, os ) ;
  ecrireTexte("\r\n", os) ; // pour faire beau

  //sauver les informations (graphiques...) associées au lien
        writeAnObject.write(os) ;

        }//fin if( writeAnObject != NOOBJECT )
        else
        {//début if
        //un lien n'existe pas à cet endroit
  //sauver ce fait dans le fichier
        ecrireTexte ("Nature du lien conceptuel              H\r\n", os) ;

        }//fin if


 }//fin for

}//fin fonction

//>>        NOM DE LA METHODE:
//                TArrayWithHolesOfLinks::read(Ripstream )
Pvoid TArrayWithHolesOfLinks::read(Ripstream is)
{//début fonction
TFenPrinc *pFenPrinc =
                        (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;

char                    cTypeEntreeTableauLien ;
unsigned int    nLnNomClasseOR,nLnNomClasseEXTR ,nLastIndex;
```

```
char                    szClassNameOR[LGMAXNOM],
                                szClassNameEXTR[LGMAXNOM] ;

// lecture du nombre d'entrée dans la tableau de liens les trous compris
// qui corresponde à lastElementIndex+1 car notre lowerbound commence à 0

int nLongueur = strlen
                        ("Nature du lien conceptuel      :     H\r\n") ;
                         // 45 repérés !!
                         // c'est parce que ces chaines écrites sont de même
                         // longueur qu'on ne crée qu'une seule variable nLongueur

char    * szTampon = new char [nLongueur + 1] ;
char    szHouL [4] ;
char    szNombre [4] ;
int     NbreCar = 3 ; // correspond à %3d
int     PosInit = nLongueur - NbreCar - 2 ;
                                // 45     - 3     - 1
                                // = 41 cad le 42ième
                                // en partant de Zéro

        lire (szTampon,nLongueur, is) ;
        extraire (szTampon,PosInit,NbreCar) ;

        nLastIndex = atoi (szTampon) ;

        //is >> nLastIndex ;
        lastElementIndex = nLastIndex - 1 ;
        if( lastElementIndex > upperbound )
                reallocate(  lastElementIndex - lowerbound + 1 );

        for(int IndiceOuInserer = 0 ;
          IndiceOuInserer < nLastIndex;
                        IndiceOuInserer++)
        {//début for

        lire            (szTampon, nLongueur, is) ;
        extraire        (szTampon, PosInit , NbreCar) ;
        strcpy          (szHouL, szTampon) ;

        if ( szHouL [2]== 'L') cTypeEntreeTableauLien = 'L' ;
        if ( szHouL [2]== 'H') cTypeEntreeTableauLien = 'H' ,
        if ( szHouL [2]!= 'L' && szHouL [2]!= 'H') cTypeEntreeTableauLien = '?' ;
        //      is >> cTypeEntreeTableauLien ;
        switch(cTypeEntreeTableauLien)
                {//début switch
                case  'L' :
                        // lecture du nom de la classe origine
                        // is >> nLnNomClasseOR;
           // for(int i=0;i<nLnNomClasseOR;i++)
                        //              is >> szClassNameOR[i] ;
                        // szClassNameOR[nLnNomClasseOR]=0 ;

                // lecture de la taille de la classe origine
                lire            (szTampon, nLongueur, is) ;
                extraire        (szTampon, PosInit, NbreCar) ;
                strcpy          (szNombre, szTampon) ;
                nLnNomClasseOR = atoi(szNombre) ;
    //lecture du nom de la classe origine
                lire            (szTampon, nLnNomClasseOR, is) ;
```

```
                strcpy          (szClassNameOR, szTampon) ;
                //lecture du \r\n qui suit
        lire            (szTampon,2, is) ;


                //lecture du nom de la classe extrémité
                // is >> nLnNomClasseEXTR;
                // for(i=0;i<nLnNomClasseEXTR;i++)
                //                      is >> szClassNameEXTR[i] ;
                // szClassNameEXTR[nLnNomClasseEXTR]=0 ;

                // lecture de la taille de la classe extrémité
        lire            (szTampon, nLongueur, is) ;
                extraire        (szTampon, PosInit, NbreCar) ;
                strcpy          (szNombre, szTampon) ;
                nLnNomClasseEXTR = atoi(szNombre) ;
        //lecture du nom de la classe extrémité
                lire            (szTampon, nLnNomClasseEXTR, is) ;
                strcpy          (szClassNameEXTR, szTampon) ;
                //lecture du \r\n qui suit
        lire            (szTampon,2, is) ;


        // insérer le lien dans le tableau de liens ; c'est dans cette
                // méthode que la lecture des informations associées au
                // lien sont faites
                        ContenurLiensDeArrayHoles->AddLinkSansToucherDicoEntree
                                (szClassNameOR,

          szClassNameEXTR,
                                                                        pFenPrinc-
>HWindow.
                                                                        is,

          IndiceOuInserer);
          itemsInContainer++;
                        CHECK( itemsInContainer > 0 );
                        break ;
                case 'H' :
                        setData(IndiceOuInserer,&NOOBJECT) ;
                        break ;
                default :
                        break;
                }//fin switch
        }//fin for
delete szTampon ;
return this ;


}//fin fonction
//                              ++++++++++++++++++PUBLIQUE+++++++++++++++
//[]-> Methodes d'information
//-------------
//->        NOM DE LA METHODE
//TTableauProprietesDsDom::PropDsDomEstMembre


BOOL TTableauProprietesDsDom::PropDsDomEstMembre(
        TCoupleIndiceDsTabEtBoolOpt* ,char * szNomclasseDom,
        char * szName, HWND hwindDsDom)
```

```
{//début  fonction
TPropriete  * pPrpDsDom = new  TPropriete(szName,0);

 if(hasMember(*pPrpDsDom) )
 {//début if

        int resIndice =TrouverIndiceDsTableau(*pPrpDsDom) ;
        char bufPropDsDomEstMembre[200] ;
        strcpy(bufPropDsDomEstMembre,"le lien statique '") ;
        strcat(bufPropDsDomEstMembre,szNomclasseDom);
        strcat(bufPropDsDomEstMembre, " possède une propriété a valeur dans le domaine");
        strcat(bufPropDsDomEstMembre,
         ((TPropriete&)(*this)[resIndice]).GetNomDomaine());
        strcat(bufPropDsDomEstMembre," et portant le nom \"");
        strcat(bufPropDsDomEstMembre,
         ((TPropriete&)(*this)[resIndice]) .GetNomPropriete());
         strcat(bufPropDsDomEstMembre,"\"");
        strcat(bufPropDsDomEstMembre,
        "'\n est déjà présent dans le tableau de propriétés") ;
   for(int i = 0; i<=3;i++) MessageBeep(MB_ICONHAND) ;
        MessageBox(hwindDsDom,bufPropDsDomEstMembre," O Etoile ",MB_ICONSTOP|MB_OK)
;

        delete  pPrpDsDom ;
        return TRUE ;

 }//fin if(hasMember(*pPrpDsDom) )
 delete  pPrpDsDom ;
 return FALSE ;


}//fin fonction
//>>        NOM DE LA METHODE:
//                TTableauProprietesDsDom:: read(Ripstream)
Pvoid TTableauProprietesDsDom::read(Ripstream is)
{//début fonction
TFenPrinc *pFenPrinc =
                  (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;

char                    cTypeEntreeTableauLien ;
unsigned int    nLnNomPropriete,
                        nLastIndex,
                        IndDomaine;
char                    szNomProp[LGMAXNOM];

// lecture du nombre d'entrées dans la tableau de liens les trous
// compris qui correspondent à lastElementIndex+1 car notre
// lowerbound commence à 0

int nLongueur = strlen (
                                                "Nombre d'entrées Tableau de liens  :   %3d\r\n"
                                                ) ; // 45 repérés !!
                        // c'est parce que ces chaines écrites sont de même
                        // longueur qu'on ne crée qu'une seule variable nLongueur

char    * szTampon = new char [nLongueur + 1],
                szIndiceDomaine [4] ;
char    szPouH [4] ;
int     NbreCar = 3 ; // correspond à %3d
int     PosInit = nLongueur - NbreCar - 2 ;
```

```
                                // 45     - 3     - 1
                                // = 41 cad le 42ième
                                // en partant de Zéro

    lire            (szTampon,nLongueur, is) ;
    extraire        (szTampon,PosInit,NbreCar) ;

    nLastIndex = atoi (szTampon) ;
    lastElementIndex = nLastIndex - 1 ;

    if( lastElementIndex > upperbound )
                    reallocate(  lastElementIndex - lowerbound + 1 );
for(int IndiceOuInserer = 0 ;
      IndiceOuInserer < nLastIndex;
                IndiceOuInserer++)
    {//début for
    lire            (szTampon, nLongueur, is) ;
    extraire        (szTampon, PosInit , NbreCar) ;
    strcpy          (szPouH, szTampon) ;

    if ( szPouH [2]== 'P') cTypeEntreeTableauLien = 'P' ;
    if ( szPouH [2]== 'H') cTypeEntreeTableauLien = 'H' ;
    if ( szPouH [2]!= 'P' && szPouH [2]!= 'H') cTypeEntreeTableauLien = '?' .

            // is >> cTypeEntreeTableauLien ,
            switch(cTypeEntreeTableauLien)
            {//début switch
            case  'P' :
            //lecture de la longueur du nom de propriété
            lire            (szTampon, nLongueur, is) ;
            extraire (szTampon,PosInit,NbreCar) ;
  nLnNomPropriete = atoi (szTampon) ;

            // lecture du nom de la propriété
            lire            (szTampon, nLnNomPropriete, is) ;
            strcpy          (szNomProp, szTampon);

            // lecture des caractères \r\n pour faire beau
            lire            (szTampon, 2, is) ;

  // lecture de l'indice du domaine
            lire            (szTampon, nLongueur, is) ;
            extraire        (szTampon, PosInit, NbreCar) ;
            strcpy          (szIndiceDomaine, szTampon);
  IndDomaine = atoi(szIndiceDomaine) ;

  // construction d'une propriété
            TPropriete * pPropDsDom =
                                    new TPropriete(szNomProp,IndDomaine) ;

            setData(IndiceOuInserer,pPropDsDom) ;

            itemsInContainer++,
        CHECK( itemsInContainer > 0 );
                    break ;
            case 'H'  :
              setData(IndiceOuInserer,&NOOBJECT) ;
      break ;
            default :
            char Tmp [30] ;
```

```
                wsprintf(Tmp, "CARACTERE ni H ni P . %c !!!",
                                cTypeEntreeTableauLien) ;
                MessageBox(pFenPrinc->HWindow,
                                Tmp,
                                "Problème de lecture",MB_OK);
                  break;
                }//fin switch

        }//fin for
delete szTampon ;
return this ;
}//fin fonction


//>>        NOM DE LA METHODE:
//                TTableauProprietesDsDom:: write(Ropstream )
void TTableauProprietesDsDom::write(Ropstream os)
{//début fonction
// sauver le nombre d'entrée dans la tableau de liens les trous compris
// qui correspond à lastElementIndex+1 car notre lowerbound commence à 0

        ecrire ("Nombre d'entrées Tableau de liens  :    %3d\r\n",
                        lastElementIndex + 1,
                        os ) ;

        for(int IndiceLien = 0 ;
                          IndiceLien < (lastElementIndex + 1 );
                    IndiceLien++)
        {//début for
        RTPropriete writeAnObject = (RTPropriete) objectAt(IndiceLien) ;

        if ( writeAnObject != NOOBJECT )
        {//début if
                // un lien existe à cet endroit
                // sauver ce fait dans le fichier à des fins de relectures
                ecrireTexte ("Type de lien du Tableau Propriétés :      P\r\n",
                        os ) ;
                // sauver le nom de la propriété
                char tmpNomPropriete[LGMAXNOM] ;
                lstrcpy(tmpNomPropriete,writeAnObject.GetNomPropriete() ) ;

                ecrire ("Taille en car. de la Propriété     :    %3d\r\n",
                                lstrlen(tmpNomPropriete),
                                os ) ;
                ecrireTexte (tmpNomPropriete, os ) ;
                ecrireTexte ("\r\n", os) ; // pour faire beau
                ecrire ("Indice du domaine de Propriété     :    %3d\r\n",
                                (int)(writeAnObject.GetIndiceDomaine()),
                                os ) ;
        }//fin if( writeAnObject != NOOBJECT )
        else
                {//début if
                //un lien n'existe pas à cet endroit
      //sauver ce fait dans le fichier
                ecrireTexte ("Type de lien du Tableau Propriétés :      H\r\n",
                        os ) ;

      }//fin if
  }//fin for
}//fin fonction
```

```
//>>        NOM DE LA METHODE:
//                TTableauDomaines:: read(Ripstream)
Pvoid TTableauDomaines::read(Ripstream is)
{//début fonction
TFenPrinc *pFenPrinc =
                          (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
char                      cTypeEntreeTableauLien ;

char                      szHouD [4] ;
unsigned int     nLnNomDomaine,
                          nLnExpression,
                          nLastIndex;
char                      szNomDomaine[LGMAXNOM],
                          szExpresDom[LGMAXEXPR];

// nLongueur vaut pour les 3 chaines dans la fonction membre
// le total nLongueur vaut toujours 45 - attention car \r\n vaut
// ici pour 2 caractères
int                       nLongueur = strlen
                                  ("Nbre Entrées du tabl. de domaines      %3d\r\n") ;

char                      * szTampon              = new char [nLongueur + 1],
                          * szTamponExp           = new char [LGMAXEXPR],
                          szNombre [4] ;
int                       NbCar                   = 3 ;
int                       PosInit          = nLongueur - NbCar - 2 ;
                          //41ieme position, indice 40 en C++

// lecture du nombre d'entrée dans la tableau de domaines
// les trous compris qui correspondent à lastElementIndex+1
// car notre lowerbound commence à 0
        lire              (szTampon, nLongueur, is) ;
        extraire          (szTampon, PosInit , NbCar) ;
        strcpy            (szNombre, szTampon) ;
        nLastIndex = atoi(szNombre) ;

        lastElementIndex = nLastIndex - 1 ;
        if( lastElementIndex > upperbound )
                          reallocate(  lastElementIndex - lowerbound + 1 );

        for(int IndiceOuInserer = 0 ;
          IndiceOuInserer < nLastIndex;
                    IndiceOuInserer++)
        {//début for  "D" ou "H"
        lire              (szTampon, nLongueur, is) ;
        extraire          (szTampon, PosInit , NbCar) ;
        strcpy            (szHouD, szTampon) ;

        // is >> cTypeEntreeTableauLien ;
        if ( szHouD [2]== 'D') cTypeEntreeTableauLien = 'D' ;
        if ( szHouD [2]== 'H') cTypeEntreeTableauLien = 'H' ;
        if ( szHouD [2]!= 'D' && szHouD [2]!= 'H') cTypeEntreeTableauLien = '?' ;
        switch(cTypeEntreeTableauLien)
        {//début switch
        case  'D' :
                  //is >> nLnNomDomaine;
    // lecture de la taille du nom de domaine
```

```
                lire            (szTampon, nLongueur, is) ;
                extraire        (szTampon, PosInit, NbCar) ;
                strcpy          (szNombre, szTampon) ;
                nLnNomDomaine = atoi(szNombre) ;
        //lecture du nom du domaine
                lire            (szTampon, nLnNomDomaine, is) ;
                strcpy          (szNomDomaine, szTampon) ;
                //lecture du \r\n qui suit
        lire            (szTampon,2, is) ;

                //is >> nLnExpression;
        // lecture de la taille de l'expression du domaine
                lire            (szTampon, nLongueur, is) ;
                extraire        (szTampon, PosInit,NbCar) ;
                strcpy          (szNombre, szTampon) ;
                nLnExpression = atoi(szNombre) ;
        // lecture de l'expression du domaine
                lire            (szTamponExp, nLnExpression, is) ;
                strcpy          (szExpresDom, szTamponExp) ;
                //lecture du \r\n qui suit
        lire            (szTampon,2, is) ;

                // création du domaine
                TDomaine * pDom = new TDomaine(szNomDomaine,szExpresDom) ;

                setData(IndiceOuInserer,pDom) ;

                itemsInContainer++;
           CHECK( itemsInContainer > 0 );
                  break ;
          case 'H'  :
                setData(IndiceOuInserer,&NOOBJECT) ,
         break ;
          default :
                char Tmp [60] ;
                wsprintf(Tmp, "Domaine cTypeEntreeTableauLien (D ou H ) = -%3s-",
                        szHouD) ;

                MessageBox(pFenPrinc->HWindow,Tmp,"UNKNOWN CHARACTER",MB_OK);

                        break;
          }//fin switch

          }//fin for
delete szTampon, szTamponExp ;
return this ;
}//fin fonction

//>>        NOM DE LA METHODE:
//              TTableauDomaines:: write(Ropstream )
void TTableauDomaines::write(Ropstream os)
{//début fonction
//sauver le nombre d'entrée dans la tableau de domaines les trous compris
//qui correspond à lastElementIndex+1 car notre lowerbound commence à 0
          ecrire ("Nbre Entrées du tabl. de domaines  :    %3d\r\n",
                        lastElementIndex + 1,
                        os) ;

          for(int IndiceLien = 0 ;
                IndiceLien < (lastElementIndex + 1 );
```

```
    IndiceLien++)
            {//début for
            RTDomaine writeAnObject =
                                        (RTDomaine) objectAt(IndiceLien) ;

            if ( writeAnObject != NOOBJECT )
            {//début if
            //un lien existe à cet endroit
            //sauver ce fait dans le fichier à des fins de relectures
            ecrireTexte("Nature du lien existant        :    D\r\n",
                                os) ;
            //sauver le nom de la classe origine
            char tmpNomDomaine[LGMAXNOM],tmpExprDomaine[LGMAXEXPR]
            lstrcpy(tmpNomDomaine,writeAnObject.GetNomDomaine()) ;

            ecrire ("Longueur du nom du domaine        :    %3d\r\n",
                        lstrlen(tmpNomDomaine),
        os) ;

            ecrire (tmpNomDomaine,lstrlen(tmpNomDomaine), os) ;
    // pour une belle présentation on écrit \r\n
            ecrireTexte ("\r\n", os) ;

            lstrcpy(tmpExprDomaine,writeAnObject.GetNomExprDomaine()) ;
            ecrire ("Longueur Expression du domaine    :    %3d\r\n",
                        lstrlen(tmpExprDomaine),
                        os) ;

            ecrire (tmpExprDomaine,lstrlen(tmpExprDomaine),      os) ;
            // pour une belle présentation on écrit \r\n
            ecrireTexte ("\r\n", os) ;

            }//fin if( writeAnObject != NOOBJECT )
            else
                    {//début if
                    //un lien n'existe pas à cet endroit
                    //sauver ce fait dans le fichier
                    ecrireTexte
                                ("Nature du lien existant        :    H\r\n",
                                os) ;
            }//fin if
        }//fin for
}//fin fonction

//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE:TConteneurLiensHERIT
//*************************************************************************

//                          ++++++++++++++++++PUBLIQUE+++++++++++++++
//[]-> Methodes d'information
//-------------
//->        NOM DE LA METHODE:
//TConteneurLiensHERIT:: LienEstMembre
//
BOOL  TConteneurLiensHERIT:: LienEstMembre
            (TSchemaClasse * pClassOrig,TSchemaClasse *pClassExtrem,
              HWND  hwndEstMembre)
{//début fonction
```

```
TLienHeritageConceptuel * pLienHerit =
          new TLienHeritageConceptuel (pClassOrig,pClassExtrem) ;

if((ptableauliens->hasMember(*pLienHerit)) )
{//début if

        char bufLienEstMembre[200] ;

        strcpy(bufLienEstMembre,"le lien statique "") ;
        strcat(bufLienEstMembre,pLienHerit->OrigineLien()->NomClasse);
        strcat(bufLienEstMembre, " hérite de ");
        strcat(bufLienEstMembre,pLienHerit->ExtremiteLien()->NomClasse);
        strcat(bufLienEstMembre,""\n est déjà présent dans le conteneur de liens") ;
  for(int i = 0; i<=3;i++) MessageBeep(MB_ICONHAND) ;
        MessageBox(hwndEstMembre,bufLienEstMembre," O Etoile ",MB_ICONSTOP|MB_OK) ;


        return TRUE ;

}//fin if((ptableauliens->hasMember(*pLienHerit)) )

return FALSE ;

}//fin fonction

//-------------
//->        NOM DE LA METHODE:
// TConteneurLiensHERIT::IsPropValeurDsClasse

BOOL  TConteneurLiensHERIT::
        IsPropValeurDsClasse(RTMessage Msg,TSchemaClasse* pSchIsProp)
{//début fonction
TListeDeLiens*  listeSuperLiens =
 LiensDtClasseEstOrigine(pSchIsProp->NomClasse,
      pSchIsProp->FenSchemaClasse->HWindow);
RContainerIterator SuperClassLiensIterator =
                                                    listeSuperLiens-> initIterator();

while ( int(SuperClassLiensIterator) != 0 )
 {//début while
 TLienHeritageConceptuel& rliencon =
                        (TLienHeritageConceptuel &)SuperClassLiensIterator++;
        if ( rliencon != NOOBJECT )
        {//début if


        if( rliencon.SuisJeCliqueDsTFenSchema(Msg) )
                {//début if
                 TSchemaClasse      * pSchemaSup = rliencon.ExtremiteLien() ;
                 strcpy(pSchIsProp->FenSchemaClasse->szSuperClasseClique,pSchemaSup-
>NomClasse);
                 pSchIsProp->FenSchemaClasse->bSuperClasseClique = TRUE ;
                 delete &SuperClassLiensIterator;
       return TRUE ;
                }//fin if


        }//fin if
 }//fin while
delete &SuperClassLiensIterator;
```

```
return FALSE;


}//fin fonction
//[]<- Methodes de manipulation
//<-        NOM DE LA METHODE:
//TConteneurLiensHERIT::
//  DetruireClasseEtDependances(LPSTR szDetDepNom,HWND hwdDetDep)
//


void TConteneurLiensHERIT::
  DetruireClasseEtDependances(LPSTR szDetDepNom,HWND hwdDetDep)

{//début fonction

//supprimer tous les liens de la classe szDetDepNom du tableau de liens
//et supprimer les indices qui référencent ces liens à partir du
//dictionnaire d'entrée
TListeDeLiens* pLiensListDetDep= LiensDeLaClasse(szDetDepNom,hwdDetDep);
RContainerIterator DetDepIterator =
                                        pLiensListDetDep-> initIterator();

while ( int(DetDepIterator) != 0 )
 {//début while
 TLienConceptuel & rlienconDepDet=(TLienConceptuel &) DetDepIterator++,
        if (rlienconDepDet != NOOBJECT )
        {//début if
          SupprimerLienConceptuel(&rlienconDepDet.hwdDetDep) ;
        }//fin if (rlienconDepDet!= NOOBJECT )
 }//fin while
delete &DetDepIterator;

//Puis détacher l'assoc corresponadnt à szDetDepNom dans le dictionnaire
//d'entrée
//le paramètre suivant ne sert pas ici;il sert quand
// TDicoEntree::RenommerClasse
//appellle TDicoEntree::DetacherClasse pour détacher l'assoc du diction
//naire mais en conservant la double liste qui servira à créer une
//nouvelle assoc qui sera inséré dans le dicoEntrée
TDeuxListsOrigExtr  * pdoubleliste ;
pdicoEntree->DetacherClasse(szDetDepNom,pdoubleliste);

}//fin fonction

//-------------
//<-        NOM DE LA METHODE:
// TConteneurLiensHERIT::AddLinkSansToucherDicoEntree
//Commentaires:
//méthode appelée à partir de TArrayWithHolesOfLinks::read une fois que
//les noms des classes origines et extrémités ont été lues à partir du
//disque; mais comme le dictionnaire d'entréé a été déjà lu les listes
//indices origines et extrémités sont correctes d'où cette deuxième
//fonction qui fait la même chose que AjouterLienHeritage sans l'appel
//de  MajListesOriginesEtExtremites
//
void TConteneurLiensHERIT::
        AddLinkSansToucherDicoEntree(LPSTR szClasseDerivee,
                                                        LPSTR szClasseDeBase,HWND
hwnd,
```

```
                                                        Ripstream is,int
OuInsererDsTableau)
{//début fonction
TSchemaClasse * pClasseOrigine ; //classe dérivée pour le moment

BOOL bOrigineCreated ;//requis pour EstMemmbre; pas nécessaire ici

TSchemaClasse * pClasseExtremite; //classe de base pour le moment

BOOL bExtremiteCreated ;//requis pour EstMemmbre ;pas nécessaire ici

int resOrigine =DicoClasses->
        EstMembre(szClasseDerivee,pClasseOrigine,bOrigineCreated,hwnd);

int resExtremite =DicoClasses->
        EstMembre(szClasseDeBase,pClasseExtremite,bExtremiteCreated,hwnd);
int resOriExt= resOrigine && resExtremite;
if( !resOriExt)
  {//début if
    char bufinfo[200]; strcpy(bufinfo,"l'une deux ou les deux classes ");
        strcat(bufinfo,szClasseDerivee) ;strcat(bufinfo," et ") ;
        strcat(bufinfo,szClasseDeBase) ;
        strcat(bufinfo,
                        " n'appartiennent pas au dictionnaire de classes ");
        MessageBeep(MB_ICONINFORMATION);
        MessageBox(hwnd,bufinfo,"Message O Etoile",

       MB_OK|MB_ICONINFORMATION) ;
        return ;
  }//fin if

 TLienHeritageConceptuel* pLienHerit =
  new TLienHeritageConceptuel(pClasseOrigine,pClasseExtremite) ;

 //on ne teste pas si le lien appartient déjà au conteneur
 //puisque cette fonction est appelée à partir de TArrayWithHolesOfLinks::read
 //et que la sauvegarde n'a pu normalement se faire qu'à partir d'une
 //structure de données correcte (cad sans doublons)
 ptableauliens->setData(OuInsererDsTableau,pLienHerit) ;

 pLienHerit->read(is);

}//fin fonction

//<-        NOM DE LA METHODE:
//TConteneurLiensHERIT::SupprimerLienHeritage(LPSTR,LPSTR,HWND)

void TConteneurLiensHERIT::SupprimerLienHeritage(LPSTR szNomClasseOrigine,
                                LPSTR szNomClasseExtremite,HWND hwind)

{//début fonction


TSchemaClasse * pClasseOrigine ; //classe dérivée pour le moment

BOOL bOrigineCreated ;//requis pour EstMemmbre; pas nécessaire ici

TSchemaClasse * pClasseExtremite; //classe de base pour le moment

BOOL bExtremiteCreated ;//requis pour EstMemmbre ;pas nécessaire ici
```

```
int resOrigine =DicoClasses->
        EstMembre(szNomClasseOrigine,pClasseOrigine,bOrigineCreated,hwind);

int resExtremite =DicoClasses->
        EstMembre(szNomClasseExtremite,pClasseExtremite,bExtremiteCreated,hwind);
int resOriExt= resOrigine && resExtremite.
if( !resOriExt)
  {//début if
  //cette alternative du if est peu probable si la présente méthode
   char bufinfo[200]; strcpy(bufinfo,"l'une deux ou les deux classes ");
        strcat(bufinfo,szNomClasseOrigine) ;strcat(bufinfo," et ") ;
        strcat(bufinfo,szNomClasseExtremite) ;
        strcat(bufinfo,
                        " n'appartiennent pas au dictionnaire de classes ");
        MessageBeep(MB_ICONINFORMATION);
        MessageBox(hwind,bufinfo,"Message O Etoile",

       MB_OK|MB_ICONINFORMATION) ;
   return ;
  }//fin if

 TLienHeritageConceptuel * pLienHerit =
 new TLienHeritageConceptuel (pClasseOrigine,pClasseExtremite) ;

 SupprimerLienConceptuel(pLienHerit ,hwind) ;


}//fin fonction
//<-        NOM DE LA METHODE:
//TConteneurLiensHERIT::VisualiserSuperClasses

void TConteneurLiensHERIT::VisualiserSuperClasses(HDC h,unsigned int HIcone,
                int ,TSchemaClasse * pSchCl)
{//début fonction

  POINT point ;
int LargTotSuperClasses=LargeurTotaleSuperClasses
                                                (
                                                h,
                                                pSchCl->NomClasse,
                                                pSchCl->FenSchemaClasse->HWindow
                                                )  ;

 //demander l'affichage du commentaire 1.5 * HIcone après celui
 //des superclasses s'il y en a des superclasses;s'il n'y a pas
 //faire l'affichage tout de suite apres le nom de la classe courante
 if(!LargTotSuperClasses)
  return ;


 TListeDeLiens*  listeLiensVisu =LiensDtClasseEstOrigine(
     pSchCl->NomClasse,pSchCl->FenSchemaClasse->HWindow).


point.y =1.2 * HIcone; // position absolue par rapport au haut
                                                // de l'écran. Les autres rubriques sont à
                // des valeurs relatives.
 point.x = 5 ;
```

```
 RContainerIterator listVisuIterator =listeLiensVisu ->initIterator();

 if( int(listVisuIterator) != 0 )
  {//début if (( int(listVisuIterator) != 0 )
        TextOut(h,point.x,point.y,
          "Classes généralisées",lstrlen("Classes généralisées") );
    point.x = 10 ;
    point.y += HIcone;//.8*HIcone;
  }//fin if(( int(listVisuIterator) != 0 )


 while ( int(listVisuIterator) != 0 )
  {//début while
 TLienHeritageConceptuel& rlienconVisu =
          (TLienHeritageConceptuel&) listVisuIterator++;
         if ( rlienconVisu!= NOOBJECT )
          {//début if

                rlienconVisu.DessineToiDsFenSchema(point,HIcone
                 /*(pSchCl->FenSchemaClasse->PointCourantAFFICHAGE).y*/,h)  ;

         }//fin if
  }//fin while
delete &listVisuIterator;
pSchCl->FenSchemaClasse->PointCourantAFFICHAGE.y =point.y +1.2 * HIcone;
// 1.2 est ici suffisant parce que le rectangle des classes généralisées
// ne contient que le nom de la classe généralisée qui est sur une seule
// ligne. Ce n'est pas le cas pour les classes référencées dont le titre
// devra se trouver à 1.2 HIcone + 3 ;
//
//  1.5*HIcone;
}//fin fonction

//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE:TConteneurLiensCOMPOS
//*************************************************************************

//++++++++-------- CONSTRUCTEURS ET DESTRUCTEUR
//-------------
//        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::TConteneurLiensCOMPOS

TConteneurLiensCOMPOS::TConteneurLiensCOMPOS()
{//début fonction
pLiensComposDeTFenSchema = new TListeDeLiens()  ;
}//fin fonction

//                                  ++++++++++++++++++++PUBLIQUE+++++++++++++++
//[]-> Methodes d'information
//-------------
//->        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::LienEstMembre

BOOL  TConteneurLiensCOMPOS::LienEstMembre
                (TSchemaClasse * pClassOrig,TSchemaClasse *pClassExtrem,
                LPSTR szNomLienCompos,
                 HWND  hwndEstMembre)
{//début fonction
```

```
TLienCompositionConceptuel* pLienCompos=
          new TLienCompositionConceptuel(pClassOrig,
               pClassExtrem,szNomLienCompos) ,

if((ptableauliens->hasMember(*pLienCompos)) )
{//début if

        char bufLienEstMembre[200] ;

        strcpy(bufLienEstMembre,"le lien statique '") ;
        strcat(bufLienEstMembre,pLienCompos->OrigineLien()->NomClasse);
        strcat(bufLienEstMembre, " est une classe composite de ");
        strcat(bufLienEstMembre,pLienCompos->ExtremiteLien()->NomClasse);
        strcat(bufLienEstMembre," et portant le nom \"");
        strcat(bufLienEstMembre,szNomLienCompos); strcat(bufLienEstMembre,"\"");
        strcat(bufLienEstMembre,"'\n est déjà présent dans le conteneur de liens") ;
  for(int i = 0; i<=3;i++) MessageBeep(MB_ICONHAND) ;
        MessageBox(hwndEstMembre,bufLienEstMembre," O Etoile ",MB_ICONSTOP|MB_OK) ;


        return TRUE ;

}//fin if((ptableauliens->hasMember(*pLienHerit)) )

return FALSE ;

}//fin fonction

//-------------
//->        NOM DE LA METHODE:
// TConteneurLiensCOMPOS::IsPropValeurDsClasse

BOOL  TConteneurLiensCOMPOS::
          IsPropValeurDsClasse(RTMessage Msg,TSchemaClasse* pSchIsProp)
{//début fonction
TListeDeLiens*  listeSuperLiens =
 LiensDtClasseEstOrigine(pSchIsProp->NomClasse,
       pSchIsProp->FenSchemaClasse->HWindow);
RContainerIterator SuperClassLiensIterator =
                                                    listeSuperLiens-> initIterator();

while ( int(SuperClassLiensIterator) != 0 )
 {//début while
 TLienCompositionConceptuel& rliencon =
                          (TLienCompositionConceptuel&)SuperClassLiensIterator++;
        if ( rliencon != NOOBJECT )
        {//début if


        if( rliencon.SuisJeCliqueDsTFenSchema(Msg) )
                  {//début if
                   pSchIsProp->FenSchemaClasse->pLienComposClique= &rliencon ,
                   pSchIsProp->FenSchemaClasse->bClasseComposanteClique= TRUE ;
                   delete &SuperClassLiensIterator;
       return TRUE ;
                  }//fin if

        }//fin if
 }//fin while
delete &SuperClassLiensIterator;
```

```
return FALSE;

}//fin fonction

//-------------
//->        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::LargeurTotalePropValDsClasse

int TConteneurLiensCOMPOS::
  LargeurTotalePropValDsClasse(HDC hDCLargTot,
                                        LPSTR szNomClasseLargTot, HWND hwindLargTot)
{//début fonction

 int LargIconeCourante,
          LargNomLienCourant,
          LargIconeEtLien,
          LargeurTotalePropDsClasse ;

 TSchemaClasse          * pSchema ;
 //initialisation à zéro
 LargeurTotalePropDsClasse = 0;

 TListeDeLiens*  listeLiensTaille =
       LiensDtClasseEstOrigine(szNomClasseLargTot,hwindLargTot);
 RContainerIterator listTailleIterator =
              listeLiensTaille ->initIterator();
 while ( int(listTailleIterator) != 0 )
  {//début while
  TLienCompositionConceptuel & rlienconTaille =
                                    (TLienCompositionConceptuel&) listTailleIterator++;
          if ( rlienconTaille!= NOOBJECT )
          {//début if
           pSchema = rlienconTaille.ExtremiteLien();

          //pSchema->FenSchemaClasse->rIconeClasse.
          //                              SetLargHautIcone(hDCLargTot,pSchema);
          //LargIconeCourante =
          //       pSchema->FenSchemaClasse->rIconeClasse.Forme.right -
          //                  pSchema->FenSchemaClasse->rIconeClasse.Forme.left ;

          HFONT pol1,pol_orig1;
          LOGFONT pol_log;

          pol_log.lfHeight = 16,
          pol_log.lfWidth =0;
          pol_log.lfItalic = 0 ;
          pol_log.lfWeight = 400 ; pol_log.lfStrikeOut=FALSE ;
          pol_log.lfUnderline = FALSE ;
          pol_log.lfCharSet = ANSI_CHARSET ;
          pol_log.lfEscapement = 0 ;pol_log.lfOrientation= 0;
          pol_log.lfOutPrecision=0;pol_log.lfClipPrecision = 0 ;
          //pol_log.lfQuality =PROOF_QUALITY ;
          pol_log.lfPitchAndFamily = FF_MODERN | FIXED_PITCH ;
          strcpy(pol_log.lfFaceName,"Courier New");

          pol1 = CreateFontIndirect(&pol_log);

          pol_orig1 = SelectObject(hDCLargTot, pol1);
          long Taille;
          Taille=GetTextExtent(hDCLargTot,
```

```
                    pSchema->NomClasse,strlen(pSchema->NomClasse) );

        SelectObject(hDCLargTot,pol_orig1);
        DeleteObject(pol1);

        LargIconeCourante=LOWORD(Taille) + 5;
        if(rlienconTaille.bTypeLien)
        {//début if(rlienconTaille.bTypeLien)
          LargIconeCourante+=5;
        }//fin  if(rlienconTaille.bTypeLien)


        HFONT pol2,pol_orig2;
        LOGFONT pol_log2 ;
   pol_log2.lfHeight = 13;
   pol_log2.lfWidth= 0 ;
   pol_log2.lfItalic = 1 ;
   pol_log2.lfWeight = 400 ; pol_log2.lfStrikeOut=FALSE ;
   pol_log2.lfUnderline = FALSE ;
   pol_log2.lfCharSet = ANSI_CHARSET ;
   pol_log2.lfEscapement = 0 ;pol_log2.lfOrientation= 0;
   pol_log2.lfOutPrecision=0;pol_log2.lfClipPrecision = 0 ;
 // pol_log2.lfQuality =DEFAULT_QUALITY ;
   pol_log2.lfPitchAndFamily = FF_DONTCARE| DEFAULT_PITCH ;
   strcpy(pol_log2.lfFaceName,"Courier New");
   pol2 = CreateFontIndirect(&pol_log2);
        pol_orig2=SelectObject(hDCLargTot, pol2);

        LargNomLienCourant = rlienconTaille.GetLargeurNomLien(hDCLargTot) ;
        SelectObject(hDCLargTot,pol_orig2);DeleteObject(pol2) ;

        if(LargIconeCourante >= LargNomLienCourant)
                LargIconeEtLien = LargIconeCourante ;
        else
                LargIconeEtLien = LargNomLienCourant ;

        LargeurTotalePropDsClasse +=LargIconeEtLien;


        }//fin if ( rlienconTaille!= NOOBJECT )
 }//fin while

 delete &listTailleIterator;

 return  LargeurTotalePropDsClasse ;

}//fin fonction

//<-        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::
//  DetruireClasseEtDependances(LPSTR szDetDepNom,HWND hwdDetDep)

void TConteneurLiensCOMPOS::
  DetruireClasseEtDependances(LPSTR szDetDepNom,HWND hwdDetDep)
{//début fonction

//supprimer tous les liens de la classe szDetDepNom du tableau de liens
//et supprimer les indices qui référencent ces liens à partir du
//dictionnaire d'entrée
TListeDeLiens* pLiensListDetDep = LiensDeLaClasse(szDetDepNom,hwdDetDep);
```

```
RContainerIterator DetDepIterator =
                                        pLiensListDetDep-> initIterator();

while ( int(DetDepIterator) != 0 )
 {//début while
 TLienConceptuel & rlienconDepDet=(TLienConceptuel &) DetDepIterator++;
        if (rlienconDepDet != NOOBJECT )
        {//début if
          SupprimerLienConceptuel(&rlienconDepDet,hwdDetDep) ;
        }//fin if (rlienconDepDet!= NOOBJECT )
 }//fin while
delete &DetDepIterator;

//Puis détacher l'assoc corresponadnt à szDetDepNom dans le dictionnaire
//d'entrée
//le paramètre suivant ne sert pas ici;il sert quand
// TDicoEntree::RenommerClasse
//appellle TDicoEntree::DetacherClasse pour détacher l'assoc du diction
//naire mais en conservant la double liste qui servira à créer une
//nouvelle assoc qui sera inséré dans le dicoEntrée
TDeuxListsOrigExtr  * pdoubleliste ;
pdicoEntree->DetacherClasse(szDetDepNom,pdoubleliste);

}//fin fonction

//[]<- Methodes de manipulation
//-------------
//<-        NOM DE LA METHODE:
// TConteneurLiensCOMPOS::AjouterLienComposition
:
void TConteneurLiensCOMPOS::
          AjouterLienComposition(TLienCompositionConceptuel* plienComp,
                HWND hwindAjouterComp)
{//début fonction
  int indTaleauLien = MettreLienDsTableauLiens(plienComp,
                    hwindAjouterComp);

  if(indTaleauLien>=0)
   {//début if
    MajListesOriginesEtExtremites
                (
                                                    (plienComp->OrigineLien())->NomClasse
                                                    (plienComp->ExtremiteLien())->NomClasse
                        indTaleauLien,
                                                    hwindAjouterComp
                                                );
        }//fin if
        else
         MessageBox(hwindAjouterComp,
           "Indice tableau liens négatif",
             "TConteneurLiensCOMPOS",MB_OK);

}//fin fonction

//<-        NOM DE LA METHODE.
// TConteneurLiensCOMPOS::AddLinkSansToucherDicoEntree

//Commentaires:
//méthode appelée à partir de TArrayWithHolesOfLinks::read une fois que
//les noms des classes origines et extrémités ont été lues à partir du
```

```
//disque; mais comme le dictionnaire d'entréé a été déjà lu les listes
//indices origines et extrémités sont correctes d'où cette deuxième
//fonction qui fait la même chose que AjouterLienHeritage sans l'appel
//de  MajListesOriginesEtExtremites
//
void TConteneurLiensCOMPOS::
        AddLinkSansToucherDicoEntree(LPSTR szClasseDerivee,
                                                        LPSTR szClasseDeBase,HWND
hwnd,
                                                        Ripstream is,int
OuInsererDsTableau)
{//début fonction
TSchemaClasse * pClasseOrigine ; //classe dérivée pour le moment

BOOL bOrigineCreated ;//requis pour EstMemmbre; pas nécessaire ici

TSchemaClasse * pClasseExtremite; //classe de base pour le moment

BOOL bExtremiteCreated ;//requis pour EstMemmbre ;pas nécessaire ici

int resOrigine =DicoClasses->
        EstMembre(szClasseDerivee,pClasseOrigine,bOrigineCreated,hwnd);

int resExtremite =DicoClasses->
        EstMembre(szClasseDeBase,pClasseExtremite,bExtremiteCreated,hwnd);
int resOriExt= resOrigine && resExtremite;
if( !resOriExt)
  {//début if
    char bufinfo[200]; strcpy(bufinfo,"l'une deux ou les deux classes ");
        strcat(bufinfo,szClasseDerivee) ;strcat(bufinfo," et ") ;
        strcat(bufinfo,szClasseDeBase) ;
        strcat(bufinfo,
                        " n'appartiennent pas au dictionnaire de classes ");
        MessageBeep(MB_ICONINFORMATION);
        MessageBox(hwnd,bufinfo,"Message O Etoile",

       MB_OK|MB_ICONINFORMATION) ;
        return ;
  }//fin if

 TLienCompositionConceptuel* pLienComposition =
  new TLienCompositionConceptuel(pClasseOrigine,pClasseExtremite) ;

 //on ne teste pas si le lien appartient déjà au conteneur
 //puisque cette fonction est appelée à partir de TArrayWithHolesOfLinks::read
 //et que la sauvegarde n'a pu normalement se faire qu'à partir d'une
 //structure de données correcte (cad sans doublons)
 ptableauliens->setData(OuInsererDsTableau,pLienComposition) ;

 pLienComposition->read(is);

}//fin fonction
//<-        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::SetDefaultsGrLinksDeTFenSch

void TConteneurLiensCOMPOS::SetDefaultsGrLinksDeTFenSch(HDC hDCSetDefault)
{//début fonction
RContainerIterator listeIteratorTFenSch
                                = pLiensComposDeTFenSchema->initIterator();
 while ( int(listeIteratorTFenSch) != 0 )
```

```
 {//début while
 TLienCompositionConceptuel & rlienconTFenSch =
                          (TLienCompositionConceptuel &)listeIteratorTFenSch++;
        if ( rlienconTFenSch!= NOOBJECT )
        {//début if

                SetDefautsGrForOneCLink( &rlienconTFenSch,hDCSetDefault) ;
        }//fin if
 }//fin while
 delete &listeIteratorTFenSch;
 //une fois établie les flèches par défaut ,on vide la liste des liens
 //sans bien sûr les détruire
 pLiensComposDeTFenSchema->ownsElements(0);
 pLiensComposDeTFenSchema->flush();


}//fin fonction
//<-        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::SetDefautsGrForOneCLink

void TConteneurLiensCOMPOS::
        SetDefautsGrForOneCLink(TLienConceptuel * pLCompos,HDC)
{//début fonction

//on établit ici une flèche arbitraire entre une icône de classe source
 // et une icône de classe extrémité; c'est une action par défaut faite
 //la TFenGrapheStatique sous les yeux; on pourra éventuellement si ce
 // lien ne convient pas le redessiner manuellement dans
 // la TFenGrapheStatique
 POINT ptOrig,ptExtrem ;
 RECT  rectOrig ,rectExtrem ;
 rectOrig =pLCompos->OrigineLien()->FenSchemaClasse->
                        rIconeClasse.Forme;
 ptOrig.x = rectOrig.right;
 ptOrig.y = rectOrig.top ;
 rectExtrem =pLCompos->ExtremiteLien()->FenSchemaClasse->
                        rIconeClasse.Forme;
 ptExtrem.x = rectExtrem.right;
 ptExtrem.y = rectExtrem.bottom ;
 pLCompos->EtablirPointsLienGraphique (ptOrig,ptExtrem,0,0 ) ;


}//fin fonction
//<-        NOM DE LA METHODE.
//TConteneurLiensCOMPOS::AjouterLienCdeTFenSCh
//
void TConteneurLiensCOMPOS::
         AjouterLienCdeTFenSCh(TLienCompositionConceptuel* pLC)
{//début fonction
if (pLC)
   pLiensComposDeTFenSchema->add(*pLC) ;

}//fin fonction

//<-        NOM DE LA METHODE
//TConteneurLiensCOMPOS::SupprimerLienComposition(LPSTR,LPSTR,HWND)

void TConteneurLiensCOMPOS::SupprimerLienComposition
        (TLienConceptuel *pLienCompos,HWND hwind)
```

```
{//début fonction

 SupprimerLienConceptuel(
 (TLienCompositionConceptuel*)pLienCompos,hwind) ;

}//fin fonction
//<-        NOM DE LA METHODE:
//TConteneurLiensCOMPOS::VisualiserPropsValDsClasse.

void TConteneurLiensCOMPOS: VisualiserPropsValDsClasse
            (HDC h,unsigned int HIcone,
                                                int , TSchemaClasse* pSch)
{//début fonction
   POINT point ;
//  point.y =1.5 * HIcone;

 //demander l'affichage du commentaire 1.5 * HIcone après celui
 //des superclasses s'il y en a des superclasses;s'il n'y a pas
 //faire l'affichage tout de suite après le nom de la classe courante
 if(!(ptableauliens->getItemsInContainer()))
        {//début if
        return ;//pas de liens de composition a afficher
        }//fin if(!getItemsInContainer())

// RECT r;
// int largeur ; GetClientRect(h, &r);largeur=r.right;//largeur maxi zone client

 TListeDeLiens*  listeLiensVisu =
            LiensDtClasseEstOrigine(pSch->NomClasse,
                                pSch->FenSchemaClasse->HWindow);
                ( listeLiensVisu->getItemsInContainer() - 1)*3  ;
 point.y =(pSch->FenSchemaClasse->PointCourantAFFICHAGE).y;
 //point.y +=1.5 * HIcone;
 point.x = 5 ;
 RContainerIterator listVisuIterator =listeLiensVisu ->initIterator();

 if( int(listVisuIterator) != 0 )
  {//début if (( int(listVisuIterator) != 0 )
    TextOut(h,point.x,point.y,
                "Classes composantes",lstrlen("Classes composantes") );
          point.x = 10 ;
    point.y += HIcone;//.8*HIcone.
  }//fin if(( int(listVisuIterator) != 0 )

 while ( int(listVisuIterator) != 0 )
  {//début while
        TLienCompositionConceptuel & rlienconVisu =
          (TLienCompositionConceptuel &) listVisuIterator++;
         if ( rlienconVisu!= NOOBJECT )
          {//début if

          rlienconVisu.DessineToiDsFenSchema(point,HIcone,h) ;

        }//fin if
  }//fin while
delete &listVisuIterator;
pSch->FenSchemaClasse->PointCourantAFFICHAGE.y  = point.y + 1.2 * HIcone

}//fin fonction
//-------------
```

```
//<< >> << >> << >>Methode de gestion des flux

//***************************************************************************
//***************************************************************************
//              METHODES DE LA CLASSE:TConteneurLiensHERITCOMPOS
//***************************************************************************

//++++++++-------- CONSTRUCTEURS ET DESTRUCTEUR
//-------------
//        NOM DE LA METHODE:
//TConteneurLiensHERITCOMPOS::TConteneurLiensHERITCOMPOS

TConteneurLiensHERITCOMPOS::TConteneurLiensHERITCOMPOS()
{//début fonction
CopierConteneurLiensHeritages();
CopierConteneurLiensCompositions();
}//fin fonction
//-------------
//        NOM DE LA METHODE:
//TConteneurLiensHERITCOMPOS::~TConteneurLiensHERITCOMPOS

TConteneurLiensHERITCOMPOS::~TConteneurLiensHERITCOMPOS()
{//début fonction
//DetruireCopieLiensHeritages();
//DetruireCopieLiensCompositions();
//delete pdicoEntree;
//delete ptableauliens;

}//fin fonction
//                              ++++++++++++++++++PUBLIQUE+++++++++++++++
//[]-> Methodes d'information
//-------------
//[]<- Methodes de manipulation
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiensHERITCOMPOS::CopierConteneurLiensHeritages

void TConteneurLiensHERITCOMPOS::CopierConteneurLiensHeritages()
{//début fonction

TFenPrinc *pFenPrinc =
                        (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
//recopier le dictionnaire d'entrée du conteneur de liens d'héritage
//dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graphes Héritage et Composition
RContainerIterator dicoIteratorHERIT = ConteneurLiensHERIT->
                                                                pdicoEntree-
>initIterator();

 while ( int(dicoIteratorHERIT) != 0 )
 {//début while

        RTNomEtListeDoubleAssoc dicoAnObjectHERIT=
                                        (RTNomEtListeDoubleAssoc)dicoIteratorHERIT++
        if ( dicoAnObjectHERIT != NOOBJECT )
        {//début if
        String    *str =  new String("");
                        *str  =  ( String & ) ( dicoAnObjectHERIT .key() ) ;
        //créé paire listes vides
```

```
        TDeuxListsOrigExtr *listsDE =
                        new TDeuxListsOrigExtr() ;
        //crée l'assoc (nomClasse,paire liste vide)
        TNomEtListeDoubleAssoc *entreeDE =
                        new TNomEtListeDoubleAssoc(*str,*listsDE) ;
        //ajouter l'assoc dans DicoEntree
        pdicoEntree->add(*entreeDE) ;

        }//fin if ( cyclesAnObjectHERIT!= NOOBJECT )
 }//fin while
 delete &dicoIteratorHERIT ;

//recopier le tableau liens du conteneur de liens d'héritage
//dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graphes Héritage et Composition

RContainerIterator tableaulteratorHERIT = ConteneurLiensHERIT->
                                                        ptableauliens->initIterator();

 while ( int(tableaulteratorHERIT) != 0 )
 {//début while

        RTLienHeritageConceptuel tableauAnObjectHERIT=

(RTLienHeritageConceptuel)tableaulteratorHERIT++;
        if ( tableauAnObjectHERIT!= NOOBJECT )
  {//début if
                TLienHeritComposConcept *pLienHeritCompos
                        = new TLienHeritComposConcept(
                        tableauAnObjectHERIT. OrigineLien() ,
                        tableauAnObjectHERIT.ExtremiteLien () ) ;
                  pLienHeritCompos->SetNatureDuLien(HERIT) ;
                int indTaleauLien =
                        MettreLienDsTableauLiens(
                                (PTLienHeritageConceptuel)pLienHeritCompos,
                                        pFenPrinc->HWindow);


          if(indTaleauLien>=0)
    {//début if
                  MajListesOriginesEtExtremites
                                                (
                   (tableauAnObjectHERIT.OrigineLien())->NomClasse ,
                        (tableauAnObjectHERIT ExtremiteLien())->NomClasse,
                        indTaleauLien,
                                                        pFenPrinc->HWindow
                                                );
          }//fin if (indTaleauLien>=0)

        }//fin if( tableauAnObjectHERIT != NOOBJECT )
 }//fin while
 delete &tableaulteratorHERIT;

}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiensHERITCOMPOS::CopierConteneurLiensCompositions

void TConteneurLiensHERITCOMPOS::CopierConteneurLiensCompositions()
{//début fonction
```

```
TFenPrinc *pFenPrinc =
                        (( TFenPrinc *)(GetApplicationObject()->MainWindow)) ;
//recopier le dictionnaire d'entrée du conteneur de liens de composition
//dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graphes Héritage et Composition
RContainerIterator dicoIteratorCOMPOS = ConteneurLiensCOMPOS->
                                                    pdicoEntree->initIterator();

 while ( int(dicoIteratorCOMPOS) != 0 )
 {//début while

        RTNomEtListeDoubleAssoc dicoAnObjectCOMPOS=

(RTNomEtListeDoubleAssoc)dicoIteratorCOMPOS++,
        if ( dicoAnObjectCOMPOS != NOOBJECT )
   {//début if

        String    *str =  new String("");
                        *str  =   ( String & ) ( dicoAnObjectCOMPOS.key() ) ;
        //créé paire listes vides
        TDeuxListsOrigExtr *listsDE =
                        new TDeuxListsOrigExtr() ;
        //crée l'assoc (nomClasse,paire liste vide)
        TNomEtListeDoubleAssoc *entreeDE =
                        new TNomEtListeDoubleAssoc(*str,*listsDE) ;
        //ajouter l'assoc dans DicoEntree
        pdicoEntree->add(*entreeDE) ;

        }//fin if ( cyclesAnObjectCOMPOS!= NOOBJECT )
 }//fin while
 delete &dicoIteratorCOMPOS ;

//recopier le tableau liens du conteneur de liens de composition
//dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graohes Héritage et Composition

RContainerIterator tableauIteratorCOMPOS = ConteneurLiensCOMPOS->
                                                    ptableauliens->initIterator();

 while ( int(tableauIteratorCOMPOS) != 0 )
 {//début while

        RTLienCompositionConceptuel tableauAnObjectCOMPOS=
                (RTLienCompositionConceptuel ) tableauIteratorCOMPOS++;
        if ( tableauAnObjectCOMPOS!= NOOBJECT )
  {//début if
                TLienHeritComposConcept *pLienHeritCompos
                        = new TLienHeritComposConcept(
                        tableauAnObjectCOMPOS. OrigineLien() ,
                        tableauAnObjectCOMPOS.ExtremiteLien (),
                        tableauAnObjectCOMPOS.NomLienComposition()      ) ,
                    pLienHeritCompos->SetNatureDuLien(COMPOS) ;
                    pLienHeritCompos->SetTypeLien(
                          tableauAnObjectCOMPOS.TypeLien(),
                          tableauAnObjectCOMPOS.GetTypeDelaMultiplicite()
                       );

                int indTaleauLien =
                        MettreLienDsTableauLiens(
```

```
                                (PTLienCompositionConceptuel)pLienHeritCompos,
                                        pFenPrinc->HWindow);

        if(indTaleauLien>=0)
    {//début if
                MajListesOriginesEtExtremites
                                                (
                    (tableauAnObjectCOMPOS.OrigineLien())->NomClasse ,
                    (tableauAnObjectCOMPOS.ExtremiteLien())->NomClasse,
                    indTaleauLien,
                                                        pFenPrinc->HWindow
                                                );
        }//fin if (indTaleauLien>=0)

      }//fin if( tableauAnObjectCOMPOS != NOOBJECT )
 }//fin while
 delete &tableaulteratorCOMPOS
}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiensHERITCOMPOS::DetruireCopieLiensHeritages


void TConteneurLiensHERITCOMPOS::DetruireCopieLiensHeritages()
{//début fonction

//détruire la copie du dictionnaire d'entrée du conteneur de liens d'héritage
//présente dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graphes Héritage et Composition
RContainerIterator dicolteratorHERIT = pdicoEntree->initIterator();

 while ( int(dicolteratorHERIT) != 0 )
 {//début while

      RTNomEtListeDoubleAssoc dicoAnObjectHERIT=
                                                (RTNomEtListeDoubleAssoc)dicolteratorHERIT++;
      if ( dicoAnObjectHERIT != NOOBJECT )
      {//début if

      delete  &((String&) dicoAnObjectHERIT.key()) ;
      delete  &((TDeuxListsOrigExtr&) dicoAnObjectHERIT.value()) ;
      delete  &dicoAnObjectHERIT ;

      }//fin if ( cyclesAnObjectHERIT!= NOOBJECT )
 }//fin while
 delete &dicolteratorHERIT ;

//détruire la copie du tableau liens du conteneur de liens d'héritage
//présent dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graphes Héritage et Composition

RContainerIterator tableaulteratorHERIT = ptableauliens->initIterator();

 while ( int(tableaulteratorHERIT) != 0 )
 {//début while

      RTLienHeritageConceptuel tableauAnObjectHERIT=

(RTLienHeritageConceptuel)tableaulteratorHERIT++;
      if ( tableauAnObjectHERIT!= NOOBJECT )
```

```
        {//début if

        delete  &tableauAnObjectHERIT ;


        }//fin if( tableauAnObjectHERIT != NOOBJECT )
 }//fin while
 delete &tableaulteratorHERIT;


}//fin fonction
//-------------
//<-        NOM DE LA METHODE:
//TConteneurLiensHERITCOMPOS::DetruireCopieLiensCompositions

void TConteneurLiensHERITCOMPOS::DetruireCopieLiensCompositions()
{//début fonction
//détruire  la copie du dictionnaire d'entrée du conteneur de liens de composition
//présente dans  celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graphes Héritage et Composition
RContainerIterator dicolteratorCOMPOS = pdicoEntree->initIterator();

 while ( int(dicolteratorCOMPOS) != 0 )
 {//début while


        RTNomEtListeDoubleAssoc dicoAnObjectCOMPOS=

(RTNomEtListeDoubleAssoc)dicolteratorCOMPOS++;
        if ( dicoAnObjectCOMPOS != NOOBJECT )
   {//début if

        delete   &((String&) dicoAnObjectCOMPOS.key()) ;
        delete   &((TDeuxListsOrigExtr&) dicoAnObjectCOMPOS.value()) ;
        delete   &dicoAnObjectCOMPOS;

        }//fin if ( cyclesAnObjectCOMPOS!= NOOBJECT )
 }//fin while
 delete &dicolteratorCOMPOS ;

//détruire la copie du tableau liens du conteneur de liens de composition
//présente dans celui du conteneur fictif servant à tester l'existence de
// cycles mixtes dans les graohes Héritage et Composition

RContainerIterator tableaulteratorCOMPOS = ptableauliens->initIterator();

 while ( int(tableaulteratorCOMPOS) != 0 )
 {//début while

        RTLienCompositionConceptuel tableauAnObjectCOMPOS=
                (RTLienCompositionConceptuel ) tableaulteratorCOMPOS++;
        if ( tableauAnObjectCOMPOS!= NOOBJECT )
        {//début if

        delete  &tableauAnObjectCOMPOS;

        }//fin if( tableauAnObjectCOMPOS != NOOBJECT )
 }//fin while
 delete &tableaulteratorCOMPOS;
```

```
}//fin fonction

//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE: TConteneurLiensREFER
//*************************************************************************

//++++++++-------- CONSTRUCTEURS ET DESTRUCTEUR
//-------------
//        NOM DE LA METHODE:
//TConteneurLiensREFER::TConteneurLiensREFER

TConteneurLiensREFER::TConteneurLiensREFER()
{//début fonction
pLiensReferDeTFenSchema = new TListeDeLiens()  ;
}//fin fonction

//[]-> Methodes d'information
//-------------
//->        NOM DE LA METHODE:
//TConteneurLiensREFER::LienEstMembre

BOOL  TConteneurLiensREFER::LienEstMembre
                (TSchemaClasse * pClassOrig,TSchemaClasse *pClassExtrem,
                LPSTR szNomLienRefer,
                  HWND  hwndEstMembre)
{//début fonction

TLienReferenceConceptuel* pLienRefer =
          new TLienReferenceConceptuel(pClassOrig,
                          pClassExtrem,szNomLienRefer) ;

if((ptableauliens->hasMember(*pLienRefer)) )
{//début if

        char bufLienEstMembre[200] ;

        strcpy(bufLienEstMembre,"le lien statique '") ;
        strcat(bufLienEstMembre,pLienRefer->OrigineLien()->NomClasse);
        strcat(bufLienEstMembre, " est une classe composite de ");
        strcat(bufLienEstMembre,pLienRefer->ExtremiteLien()->NomClasse);
        strcat(bufLienEstMembre," et portant le nom \"");
        strcat(bufLienEstMembre,szNomLienRefer); strcat(bufLienEstMembre,"\"");
        strcat(bufLienEstMembre,"'\n est déjà présent dans le conteneur de liens") ;
  for(int i = 0; i<=3;i++) MessageBeep(MB_ICONHAND) ;
        MessageBox(hwndEstMembre,bufLienEstMembre," O Etoile ",MB_ICONSTOP|MB_OK) ;


        return TRUE ;

}//fin if((ptableauliens->hasMember(*pLienRefer)) )

return FALSE ;

}//fin fonction


//->        NOM DE LA METHODE
//  TConteneurLiensREFER::IsPropValeurDsClasse
```

```
BOOL  TConteneurLiensREFER::
          IsPropValeurDsClasse(RTMessage Msg,TSchemaClasse* pSchIsProp)
{//début fonction
TListeDeLiens*  listeReferLiens = LiensDtClasseEstOrigine(pSchIsProp->NomClasse,
        pSchIsProp->FenSchemaClasse->HWindow);
RContainerIterator ReferClassLiensIterator = listeReferLiens-> initIterator();

while ( int(ReferClassLiensIterator) != 0 )
 {//début while
 TLienReferenceConceptuel& rliencon =
                          (TLienReferenceConceptuel&)ReferClassLiensIterator++;
         if ( rliencon != NOOBJECT )
         {//début if

         if( rliencon.SuisJeCliqueDsTFenSchema(Msg) )
                  {//début if
                   pSchIsProp->FenSchemaClasse->pLienReferClique= &rliencon ;
                   pSchIsProp->FenSchemaClasse->bClasseReferenceeClique= TRUE ;
                   delete &ReferClassLiensIterator;
        return TRUE ;
                  }//fin if

         }//fin if
 }//fin while
delete &ReferClassLiensIterator;
return FALSE;

}//fin fonction

//-------------
//->        NOM DE LA METHODE:
//TConteneurLiensREFER::LargeurTotalePropValDsClasse

int TConteneurLiensREFER::
  LargeurTotalePropValDsClasse(HDC hDCLargTot,
                                        LPSTR szNomClasseLargTot, HWND hwindLargTot)
{//début fonction

 int LargIconeCourante,
         LargNomLienCourant,
         LargIconeEtLien,
         LargeurTotalePropDsClasse ;

 TSchemaClasse         * pSchema ;
 //initialisation à zéro
 LargeurTotalePropDsClasse = 0;

 TListeDeLiens*  listeLiensTaille =
       LiensDtClasseEstOrigine(szNomClasseLargTot,hwindLargTot);
 RContainerIterator listTailleIterator =
              listeLiensTaille ->initIterator();
 while ( int(listTailleIterator) != 0 )
  {//début while
  TLienReferenceConceptuel& rlienconTaille =
                                  (TLienReferenceConceptuel&) listTailleIterator++;
         if ( rlienconTaille!= NOOBJECT )
         {//début if
          pSchema = rlienconTaille.ExtremiteLien();

         //pSchema->FenSchemaClasse->rIconeClasse.
```

```
//                          SetLargHautIcone(hDCLargTot,pSchema);
//LargIconeCourante =
//      pSchema->FenSchemaClasse->rIconeClasse.Forme.right -
//                  pSchema->FenSchemaClasse->rIconeClasse.Forme.left ;

HFONT pol1,pol_orig1;
LOGFONT pol_log;

pol_log.lfHeight = 16;
pol_log.lfWidth =0;
pol_log.lfItalic = 0 ;
pol_log.lfWeight = 400 ; pol_log.lfStrikeOut=FALSE ;
pol_log.lfUnderline = FALSE ;
pol_log.lfCharSet = ANSI_CHARSET ;
pol_log.lfEscapement = 0 ;pol_log.lfOrientation= 0;
pol_log.lfOutPrecision=0;pol_log.lfClipPrecision = 0 ;
//pol_log.lfQuality =PROOF_QUALITY ;
pol_log.lfPitchAndFamily = FF_MODERN | FIXED_PITCH ;
strcpy(pol_log.lfFaceName,"Courier New");

pol1 = CreateFontIndirect(&pol_log);

pol_orig1 = SelectObject(hDCLargTot, pol1);
long Taille;
Taille=GetTextExtent(hDCLargTot,
                  pSchema->NomClasse,strlen(pSchema->NomClasse) );

SelectObject(hDCLargTot,pol_orig1);
DeleteObject(pol1);

LargIconeCourante=LOWORD(Taille) + 5;
if(rIenconTaille.bTypeLien)
{//début if(rIenconTaille.bTypeLien)
  LargIconeCourante+=5;
}//fin  if(rIenconTaille.bTypeLien)


HFONT pol2,pol_orig2;
LOGFONT pol_log2 ;
pol_log2.lfHeight = 13;
pol_log2.lfWidth= 0 ;
pol_log2.lfItalic = 1 ;
pol_log2.lfWeight = 400 ; pol_log2.lfStrikeOut=FALSE ;
pol_log2.lfUnderline = FALSE ;
pol_log2.lfCharSet = ANSI_CHARSET ;
pol_log2.lfEscapement = 0 ;pol_log2.lfOrientation= 0;
pol_log2.lfOutPrecision=0;pol_log2.lfClipPrecision = 0 ;
// pol_log2.lfQuality =DEFAULT_QUALITY ;
pol_log2.lfPitchAndFamily = FF_DONTCARE| DEFAULT_PITCH ;
strcpy(pol_log2.lfFaceName,"Courier New");
pol2 = CreateFontIndirect(&pol_log2);
pol_orig2=SelectObject(hDCLargTot, pol2);

LargNomLienCourant = rIenconTaille.GetLargeurNomLien(hDCLargTot) ;
SelectObject(hDCLargTot.pol_orig2);DeleteObject(pol2) ;

if(LargIconeCourante >= LargNomLienCourant)
        LargIconeEtLien = LargIconeCourante ;
else
        LargIconeEtLien = LargNomLienCourant ;
```

```
        LargeurTotalePropDsClasse +=LargIconeEtLien;


        }//fin if ( rlienconTaille!= NOOBJECT )
 }//fin while

 delete &listTailleIterator;

 return  LargeurTotalePropDsClasse ;

}//fin fonction
//<-        NOM DE LA METHODE
//TConteneurLiensREFER::
//  DetruireClasseEtDependances(LPSTR szDetDepNom,HWND hwdDetDep)

void TConteneurLiensREFER::
  DetruireClasseEtDependances(LPSTR szDetDepNom,HWND hwdDetDep)

{//début fonction

//supprimer tous les liens de la classe szDetDepNom du tableau de liens
//et supprimer les indices qui référencent ces liens à partir du
//dictionnaire d'entrée
TListeDeLiens* pLiensListDetDep = LiensDeLaClasse(szDetDepNom,hwdDetDep);
RContainerIterator DetDepIterator =
                                        pLiensListDetDep-> initIterator();

while ( int(DetDepIterator) != 0 )
 {//début while
 TLienConceptuel & rlienconDepDet=(TLienConceptuel &) DetDepIterator++;
        if (rlienconDepDet != NOOBJECT )
        {//début if
          SupprimerLienConceptuel(&rlienconDepDet,hwdDetDep) ;
        }//fin if (rlienconDepDet!= NOOBJECT )
 }//fin while
delete &DetDepIterator;

//Puis détacher l'assoc corresponadnt à szDetDepNom dans le dictionnaire
//d'entrée
//le paramètre suivant ne sert pas ici;il sert quand
// TDicoEntree::RenommerClasse
//appellle TDicoEntree::DetacherClasse pour détacher l'assoc du diction
//naire mais en conservant la double liste qui servira à créer une
//nouvelle assoc qui sera inséré dans le dicoEntrée
TDeuxListsOrigExtr  * pdoubleliste ,
pdicoEntree->DetacherClasse(szDetDepNom,pdoubleliste);

}//fin fonction

//----------
//<-        NOM DE LA METHODE.
// TConteneurLiensREFER::AjouterLienComposition

void TConteneurLiensREFER::
          AjouterLienReference(TLienReferenceConceptuel* plienRef,
                HWND hwindAjouterRef)
{//début fonction
  int indTaleauLien = MettreLienDsTableauLiens(plienRef,
                        hwindAjouterRef);
```

```
  if(indTaleauLien>=0)
   {//début if
     MajListesOriginesEtExtremites
                    ( (plienRef->OrigineLien())->NomClasse,
                            (plienRef->ExtremiteLien())->NomClasse,
                           indTaleauLien,
                           hwindAjouterRef
                                                    );
          }//fin if
          else
           MessageBox(hwindAjouterRef,
             "Indice tableau liens négatif",
                     "TConteneurLiensREFER",MB_OK);

}//fin fonction

//[]<- Methodes de manipulation
//<-          NOM DE LA METHODE:
// TConteneurLiensREFER::AddLinkSansToucherDicoEntree

//Commentaires:
//méthode appelée à partir de TArrayWithHolesOfLinks::read une fois que
//les noms des classes origines et extrémités ont été lues à partir du
//disque; mais comme le dictionnaire d'entréé a été déjà lu les listes
//indices origines et extrémités sont correctes d'où cette deuxième
//fonction qui fait la même chose que AjouterLienHeritage sans l'appel
//de  MajListesOriginesEtExtremites
//
void TConteneurLiensREFER::
          AddLinkSansToucherDicoEntree(LPSTR szClasseDerivee,
                                                                  LPSTR szClasseDeBase,HWND
hwnd,
                                                                  Ripstream is,int
OuInsererDsTableau)
{//début fonction
TSchemaClasse * pClasseOrigine ; //classe dérivée pour le moment

BOOL bOrigineCreated ;//requis pour EstMemmbre; pas nécessaire ici

TSchemaClasse * pClasseExtremite; //classe de base pour le moment

BOOL bExtremiteCreated ;//requis pour EstMemmbre ;pas nécessaire ici

int resOrigine =DicoClasses->
          EstMembre(szClasseDerivee,pClasseOrigine,bOrigineCreated,hwnd);

int resExtremite =DicoClasses->
          EstMembre(szClasseDeBase,pClasseExtremite,bExtremiteCreated,hwnd);
int resOriExt= resOrigine && resExtremite;
if( !resOriExt)
  {//début if
    char bufinfo[200]; strcpy(bufinfo,"l'une deux ou les deux classes ");
          strcat(bufinfo,szClasseDerivee) ;strcat(bufinfo," et ") ;
          strcat(bufinfo,szClasseDeBase) .
          strcat(bufinfo,
                          " n'appartiennent pas au dictionnaire de classes ");
          MessageBeep(MB_ICONINFORMATION);
          MessageBox(hwnd,bufinfo,"Message O Etoile",
```

```
        MB_OK|MB_ICONINFORMATION) ;
        return ;
 }//fin if

TLienReferenceConceptuel* pLienReference=
 new TLienReferenceConceptuel(pClasseOrigine,pClasseExtremite) ;

//on ne teste pas si le lien appartient déjà au conteneur
//puisque cette fonction est appelée à partir de TArrayWithHolesOfLinks::read
//et que la sauvegarde n'a pu normalement se faire qu'à partir d'une
//structure de données correcte (cad sans doublons)
ptableauliens->setData(OuInsererDsTableau,pLienReference) ;

pLienReference->read(is);

}//fin fonction

//<-        NOM DE LA METHODE:
//TConteneurLiensREFER::AjouterLienCdeTFenSCh

void TConteneurLiensREFER::
        AjouterLienRdeTFenSCh(TLienReferenceConceptuel* pLR)
{//début fonction
if (pLR)
        pLiensReferDeTFenSchema->add(*pLR) ;

}//fin fonction

//<-        NOM DE LA METHODE:
//TConteneurLiensREFER::SetDefaultsGrLinksDeTFenSch

void TConteneurLiensREFER::SetDefaultsGrLinksDeTFenSch(HDC hDCSetDefault)
{//début fonction
RContainerIterator listeIteratorTFenSch
                                = pLiensReferDeTFenSchema->initIterator();
 while ( int(listeIteratorTFenSch) != 0 )
 {//début while
 TLienReferenceConceptuel& rlienconTFenSch =
                        (TLienReferenceConceptuel&)listeIteratorTFenSch++;
        if ( rlienconTFenSch!= NOOBJECT )
        {//début if

                SetDefautsGrForOneRLink( &rlienconTFenSch,hDCSetDefault) ;
        }//fin if
 }//fin while
 delete &listeIteratorTFenSch;
 //une fois établie les flèches par défaut ,on vide la liste des liens
 //sans bien sûr les détruire
 pLiensReferDeTFenSchema->ownsElements(0);
 pLiensReferDeTFenSchema->flush();


}//fin fonction

//<-        NOM DE LA METHODE:
//TConteneurLiensREFER::SetDefautsGrForOneRLink

void TConteneurLiensREFER::
        SetDefautsGrForOneRLink(TLienConceptuel * pLRefer,HDC)
```

```
{//début fonction

//on établit ici une flèche arbitraire entre une icône de classe source
 // et une icône de classe extrémité; c'est une action par défaut faite
 //la TFenSchemaClasse sous les yeux; on pourra éventuellement si ce
 // lien ne convient pas le redessiner manuellement dans
 // la TFenGrapheStatique

 POINT ptOrig,ptExtrem ;
 RECT  rectOrig ,rectExtrem ;
 rectOrig =pLRefer->OrigineLien()->FenSchemaClasse->
                    rIconeClasse.Forme;
 ptOrig.x = rectOrig.right;
 ptOrig.y = rectOrig.top ;
 rectExtrem =pLRefer->ExtremiteLien()->FenSchemaClasse->
                     rIconeClasse.Forme;
 ptExtrem.x = rectExtrem.right;
 ptExtrem.y = rectExtrem.bottom ;
 pLRefer->EtablirPointsLienGraphique (ptOrig,ptExtrem,0,0 ) ;

}//fin fonction

//<-        NOM DE LA METHODE:
//TConteneurLiensREFER::SupprimerLienReference


void TConteneurLiensREFER::SupprimerLienReference
        (TLienConceptuel *pLienRefer,HWND hwind)

{//début fonction

  SupprimerLienConceptuel(
  (TLienReferenceConceptuel*)pLienRefer,hwind) ;

}//fin fonction

//----------
//<-        NOM DE LA METHODE:
//TConteneurLiensREFER::VisualiserPropsValDsClasse

void TConteneurLiensREFER::VisualiserPropsValDsClasse
              (HDC h,unsigned int HIcone,
                                                  int , TSchemaClasse* pSch)
{//début fonction
   POINT point ;
//  point.y =1.5 * HIcone;

  //demander l'affichage du commentaire 1.5 * HIcone après celui
  //des superclasses s'il y en a des superclasses;s'il n'y a pas
  //faire l'affichage tout de suite après le nom de la classe courante
  if(!(ptableauliens->getItemsInContainer()))
        {//début if
        return ;//pas de liens de composition a afficher
        }//fin if(!getItemsInContainer())
//  else
//        {//début else if
//        pSch->FenSchemaClasse->PointCourantAFFICHAGE.y  += HIcone ;
//        }//fin else if

 // RECT r;
```

```
// int largeur ; GetClientRect(h, &r):largeur=r.right;//largeur maxi zone client

TListeDeLiens*  listeLiensVisu =
                LiensDtClasseEstOrigine(pSch->NomClasse,
                                        pSch->FenSchemaClasse->HWindow);

 point.y =(pSch->FenSchemaClasse->PointCourantAFFICHAGE).y;
// point.y a déjà été incrémenté de 1.2 * HIcone après le "deleteobject
// il faut ajouter plus d'espace parce que le rectangle des classes
// composantes est bordé de 3 traits qui élèvent la hauteur (et la
// largeur) et qu'il contient - sous la valeur de point.y - le nom
// du lien statique dont la hauteur a été positionnée à 16.
  point.y += 16 ;
//  point.y += 0.5 * HIcone;
 point.x = 5 ;

 RContainerIterator listVisuIterator =listeLiensVisu ->initIterator();

 if( int(listVisuIterator) != 0 )
  {//début if (( int(listVisuIterator) != 0 )
    TextOut(h,point.x,point.y,
            "Classes référencées",lstrlen("Classes référencées") );
          point.x = 10 ;
    point.y += HIcone;//.8*HIcone;
  }//fin if(( int(listVisuIterator) != 0 )


 while ( int(listVisuIterator) != 0 )
  {//début while
        TLienReferenceConceptuel& rlienconVisu =
          (TLienReferenceConceptuel&) listVisuIterator++;
         if ( rlienconVisu!= NOOBJECT )
          {//début if

          rlienconVisu.DessineToiDsFenSchema(point,HIcone,h) ;

        }//fin if
 }//fin while
delete &listVisuIterator;
pSch->FenSchemaClasse->PointCourantAFFICHAGE.y  = point.y+2*HIcone ;

}//fin fonction
//-------------

//>>        NOM DE LA METHODE:
//        void ecrire(char *,int , Ropstream )

void ecrire(char *szTampon,int nValeur, Ropstream os)
{
        char szLeTampon [LGMAXEXPR + 41] ;
                                                // on fixe à 40 pour tous + 1 le
                                                // message fixe; à quoi s'ajoute
                                                // le contenu du message variable
                                                // la taille de szLeTampon doit toujours
                                                // être égale à 41 + LGMAXEXPR
                                                // \n compris
        int  nLongueur = wsprintf (szLeTampon,
                                        szTampon,
                                        nValeur
                                        );
```

```
        for (int k = 0 ; k < nLongueur ; k++)
                {
                os << szLeTampon [k] ;
                }
}

//>>        NOM DE LA METHODE:
//                void ecrireTexte (char *, Ropstream)

void ecrireTexte (char *szTampon, Ropstream os)
{
        while (*szTampon != '\0')
        os << *szTampon++ ;

}

//>>        NOM DE LA METHODE:
//                void lire(char *.Ripstream )

void lire(char *szChaine, int nLongueur, Ripstream is)
{
        for (int k = 0 ; k < nLongueur ; k++)
                {
                is >> szChaine [k] ;
                }
        szChaine [k]= 0 ;
}

//>>        NOM DE LA METHODE:
//                void extraire (char *, int, int)
//Commentaires:méthode indépendante
// copie dans cible n caractères de chaine source
// à partir d'une position donnée
//
void extraire (  char *szTamponSource,
                                int Position, int NombreCaracteres)
{
// comme la position est à la fin de la chaine, et qu'on ne
// reprend que quelques caractères, on peut déplacer les caractères
// recherchés au début de la chaine !!!
        for (int k = 0 ; k < NombreCaracteres . k++)
                {
                szTamponSource [k] = szTamponSource [Position++] ;
                }
        szTamponSource [k]= 0 ,
}

//*************************************************************************
//*************************************************************************
//              METHODES DE LA CLASSE: TEnsembleConcept
//*************************************************************************

//[]-> Methodes d'information
//-------------
//->        NOM DE LA METHODE:IdentifierConceptGr
//Commentaires:
//Cette fonction parcours un des ensembles de concepts de la classe courante
// et appelle la méthode suisjeclique pour chacun d'eux dans FenSchémaClasse
//
```

```
BOOL TEnsembleConcept::IdentifierConceptGr(POINT  posSouris,
                        TConcept * & pConceptIdentifie)


{//début fonction

RContainerIterator IterateurConcept = initIterator();

while ( int(IterateurConcept) != 0 )
 {//début while
 TConcept & rconcept =
                        (TConcept&)IterateurConcept++;
        if ( rconcept != NOOBJECT )
        {//début if
        if( rconcept.pImageSchemaClasse->SuisJeClique(posSouris) )
                {//début if
                pConceptIdentifie =   &rconcept ;
                delete &IterateurConcept;
            return TRUE ;
                }//fin if

        }//fin if
 }//fin while
delete &IterateurConcept;
return FALSE;

}//fin fonction

//[]<- Methodes de manipulation
//-------------
//**************************************************************************
//**************************************************************************
//              METHODES DE LA CLASSE: TAttribEtObjetAttribAssoc
//**************************************************************************

//                                  ++++++++++++++++++++PUBLIQUE+++++++++++++++
//-------------
//>>        NOM DE LA METHODE: streamableName
//
const Pchar TAttribEtObjetAttribAssoc::streamableName() const
{
return 'TAttribEtObjetAttribAssoc'
}
```

RÉSUME DE LA THÈSE

Le travail décrit dans le mémoire de thèse s'inscrit dans le cadre de la conception et la réalisation d'outils de support pour l'ingénierie orientée objets des besoins. L'accent est mis sur les outils de prototypage.

Un aperçu de l'état de l'art est présenté dans la première partie du mémoire. Nous suivons l'évolution des différentes méthodes d'analyse par catégorie : les méthodes cartesiennes, les methodes systémiques et les méthodes orientées objets. Nous soulignons les points forts et les points faibles de chaque catégorie et étudions les possibilités des deux premières catégories de prise en compte des avancées récentes constatées dans la méthodologie orientée objet. Une section est consacrée aux guides méthodologiques supportant la démarche d'analyse et de spécification du schéma conceptuel dans une perspective orientée objets. Notre travail s'appuie sur la méthode orientée objet O* dont le processus est basé sur le modèle en fontaine. Nous abordons l'évolution des outils CASE et soulignons les inconvénients des premiers outils CASE , notamment ceux des années 1980, à savoir leur rigidité, inflexibilité et manque d'ouverture. Nous mentionnons les solutions apportées pour resoudre ces problèmes à savoir la personnalisation, l'adaptation des outils au développeur et non l'inverse , les plates-formes d'intégration d'outils provenant de vendeurs différents. Nous présentons différentes architectures d'intégration dont PCTE.

L'ingénierie des besoins comprend les activités suivantes :
-une activité d'élicitation qui implique que l'analyste collecte de l'information sur les problèmes des clients ce qui peut être réalisé de plusieurs façons : interviews et discussions, observations étude de la documentation disponible etc. Récemment la communauté de l'ingenierie des besoins s'est considerablement focalisée sur le concept de scénario (use case). Notre outil d'élicitation est basé sur l'investigation des scénarios d'utilisation d'un système ;

-une activité de modélisation pendant laquelle l'analyste traite l'information collectée pendant l'activité precedente et met en correspondance les descriptions informelles fournies par le client avec des concepts formels dans un langage des besoins ;
-une activité de vérification dans laquelle l'analyste détecte des problèmes (contradiction, incohérences ...) dans le document des besoins ;

-une activité de validation dans laquelle l'analyste contrôle l'adéquation de ses descriptions en les reformulant de façon appropriée au client ;

Notre contribution couvre ces différents aspects. Nous proposons, en effet, les outils suivants :
-un éliciteur permettant la capture des besoins par la mise en oeuvre d'un outil de gestion de scénario utilisateur ;
-une interface d'aide à la saisie des concepts de la methode ;
-la gestion de la persistance
-un prototypeur qui réalise une application partielle (code C++) à partir des spécifications.

Summary

The work presented in the thesis is in keeping with the general pattern of tools conception and realization meant to support object oriented requirements engineering. The stress is put on prototyping tools.
A general idea of the state of the art is given in the first part of the thesis. An overview by category(i.e. cartesian, systemic and object oriented) of the evolution of the different analysis methods is presented. We underline the weaknesses and strengths of each category and study how the first two approaches can integrate the recent breakthroughs witnessed in object oriented methodology. A section is entirely devoted to methodology guidelines that support analysis and conceptual schemes specification in an object oriented perspective. Our work relies on the object oriented method O* the process of which is based on the fountain model. We tackle the evolution of CASE (Computer Aided System Engineering) tools and give the drawbacks of the former CASE tools, in particular those from the 1980s, namely rigidity, non-flexibility and lack of openness. We mention the solutions used to solve these problems, that is customization, the tools adapting themselves to the developer and not the contrary, and the CASE frameworks for tool integration. We present different integration architectures including PCTE (Portable Common Tool Environment).

Requirements engineering includes the following activities :

- an elicitation activity that involves that the analyst collects information about the client problems, that can be achieved in different manners : interviews, discussions, observations, available documentation study etc. Recently the requirements engineering community has focused considerably on the «use case» concept. Our elicitation tool is based on the investigation of the system usage scenarios ;

- a modeling activity during which the analyst processes the information obtained from the previous step and map the informal client descriptions with formal concepts of a requirements language ;

- a validation activity during which the analyst checks the adequacy of his description by re-expressing them to the client in a proper manner.

Our contribution covers these different aspects. As a matter of fact we propose the following tools :
- an elicitor thar enables the requirements capture by means of a use case management tool ;
- a persistance manager
- a prototyper that creates a partial application (in C++ code) from the specifications