

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Conception et mise en œuvre de modélisations et d'implantations de la méthode SAMPO
(Speech Act based office Modeling aPprOach)

Cravatte, Serge; Wille, Daniel

Award date:
1990

Awarding institution:
Université de Namur

[Link to publication](#)

General rights

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

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

Take down policy

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

Facultés Universitaires Notre-Dame de la Paix à Namur
Institut d'Informatique
Année académique 1989-1990

CONCEPTION ET MISE EN OEUVRE DE
MODELISATIONS ET D'IMPLANTATIONS
DE LA METHODE SAMPO

(Speech Act based office Modeling aPprOach)

Mémoire de fin d'études présenté par

Serge CRAVATTE

Daniel WILLE

pour l'obtention du grade de licencié
et maître en informatique

Promoteur : R. LESUISSE

REMERCIEMENTS

Nous sommes particulièrement heureux de remercier ici notre promoteur, Mr. Roland Lesuisse qui, tout au long de cette dernière année d'études, a suivi l'évolution de notre travail d'un oeil attentif. Ses conseils judicieux ainsi que ses remarques pertinentes ont permis à ce mémoire de devenir ce qu'il est.

Nous adressons également nos plus vifs remerciements à Mr. Kalle Lyytinen pour son accueil, le temps qu'il nous a consacré et l'aide qu'il nous a prodiguée au cours de notre stage à l'Université de Jyväskylä (Finlande). Que soient remerciés avec lui les autres membres de son équipe, avec lesquels une collaboration fructueuse a été possible. Nous pensons notamment à Mr. Esa Auramäki, à Mr. Kari Smolander et à Mr. Veli-Pekka Tahvanainen.

Nous sommes reconnaissants à Mr. Jean Vanderdonckt et à Mr. Jean-Marie Leheureux pour leur aide, souvent ponctuelle, mais précieuse, dans des domaines tant théoriques que techniques.

Enfin, nous voudrions profiter de l'occasion que nous offre ce mémoire pour exprimer notre gratitude à nos familles respectives pour le soutien et la compréhension qu'elles nous ont témoignés tout au long de nos études.

ABSTRACT

The purpose of this thesis is to present, model and implant a new information system specification method called SAMPO (Speech Act based office Modeling aPprOach), which has been developed by professor K. Lyytinen at the University of Jyväskylä, Finland.

The main feature of this method is that it focuses on the social nature of office activities and, therefore, stresses the communicative purpose that an information system has, within an organization.

Key Words : SAMPO, communication analysis method, information system specification method, model, Entity-Relationship, Object-Property-Role-Relationship, Computer Aided Software Engineering, Quickspec, implementation.

RESUME

Le but de ce mémoire est de présenter, modéliser et implanter une nouvelle méthode de spécification de systèmes d'information appelée SAMPO (Speech Act based office Modeling aPprOach), qui a été développée par le professeur K. Lyytinen à l'Université de Jyväskylä, Finlande.

La principale caractéristique de cette méthode est qu'elle met l'accent sur la nature sociale des activités de bureau et, pour ce faire, insiste sur le but communicatif que poursuit un système d'information dans une organisation.

Mots Clés : SAMPO, méthode d'analyse des communications, méthode de spécification d'un système d'information, modèle, Entité-Relation, Objet-Propriété-Rôle-Relation, Computer Aided Software Engineering, Quickspec, implantation.

TABLE DES MATIERES**INTRODUCTION****CHAPITRE 1 : PRESENTATION DE LA METHODE SAMPO**

INTRODUCTION	1.1
1.1 LA THEORIE DES ACTES DE LANGAGE	1.2
1.1.1 Les débats et enjeux	1.2
1.1.2 La notion d'acte de langage	1.3
1.1.3 Les types d'actes de langage	1.4
1.2 LA DEFINITION ET L'UTILITE DE SAMPO	1.6
1.3 L'UNIVERS DU DISCOURS	1.7
1.3.1 Le domaine entité	1.7
1.3.2 Le domaine action	1.8
1.3.2.1 Les actes instrumentaux	1.8
1.3.2.2 Les actes de langage	1.9
1.4 L'ANALYSE DU DISCOURS	1.17
1.4.1 Les concepts de l'analyse du discours	1.17
1.4.1.1 Le type de discours	1.17
1.4.1.2 La structure d'actes de langage	1.18
1.4.1.3 Les segments de discours	1.19
1.4.1.4 Les mouvements	1.19
1.4.1.5 Le tour	1.20
1.4.1.6 La topicalisation	1.20
1.4.2 Les propriétés des discours	1.21
1.4.2.1 La cohérence	1.21
1.4.2.2 La complétude	1.22
1.4.3 Les outils d'analyse du discours	1.22
1.4.3.1 Les graphes de discours	1.23
1.4.3.2 Les graphes de conversation	1.29
1.4.3.3 Les tables	1.29
CONCLUSION	1.31

CHAPITRE 2 : COMPARAISON DE QUELQUES MODELES DE STRUCTURATION DES INFORMATIONS

INTRODUCTION	2.1
2.1 LES MODELES	2.2
2.1.1 Le modèle Objet	2.2
2.1.2 Le modèle Binaire	2.3
2.1.3 Le modèle Entité-Relation (E-R)	2.4
2.1.4 Le modèle Objet-Propriété-Rôle-Relation	2.5
2.2 LE CHOIX D'UN MODELE	2.6
2.2.1 Les critères de choix	2.6
2.2.2 Le choix	2.7
2.2.2.1 Niveau 1	2.7
2.2.2.2 Niveau 2	2.8
2.2.2.3 Niveau 3	2.10
2.2.2.4 Niveau 4	2.15
2.3 LA NOTION DE META-MODELE	2.16
CONCLUSION	2.18

CHAPITRE 3 : LES OUTILS CASE SUSCEPTIBLES DE SUPPORTER UNE METHODE

INTRODUCTION	3.1
3.1 LES MODELES	3.2
3.2 LES METHODES	3.3
3.3 LES METHODOLOGIES	3.5
3.4 LES CASE TOOLKITS SPECIFIQUES ET GENERIQUES	3.7
3.5 QUELQUES CARACTERISTIQUES DES CASE TOOLKITS GENERIQUES	3.9
CONCLUSION	3.12

CHAPITRE 4 : MODELISATION ET IMPLANTATION DE LA METHODE SAMPO

INTRODUCTION	4.1
4.1 ELABORATION DU META-MODELE DE SAMPO	4.2
4.1.1 Réalisation du méta-modèle	4.2
4.1.2 Commentaires sur le méta-modèle	4.16

4.2 IMPLANTATION DE LA METHODE SAMPO	4.18
4.2.1 Introduction à un CASE toolkit générique : Quickspec	4.18
4.2.2 Présentation de MDM	4.21
4.2.3 Présentation de Quickspec	4.27
4.2.3.1 Fonctions offertes par Quickspec ...	4.27
4.2.3.2 Atouts de Quickspec	4.28
4.2.3.3 Les règles liées aux entrées	4.29
4.2.4 Réalisation de l'implantation	4.30
4.3 INTRODUCTION D'UN EXEMPLE	4.33
4.3.1 Problèmes inhérents à Quickspec	4.34
4.3.2 Réflexions sur la méthode SAMPO suite à son implantation	4.35
CONCLUSION	4.38

CHAPITRE 5 : UNE DEFINITION ET UN META-MODELE MODIFIES POUR SAMPO

INTRODUCTION	5.1
5.1 AMENAGEMENTS DE LA METHODE SAMPO ET DE SON META-MODELE	5.3
5.1.1 Le discours	5.3
5.1.2 Le domaine entité et le domaine action	5.4
5.1.3 L'entité passive	5.5
5.1.4 Les arrangements organisationnels	5.5
5.1.5 L'acte illocutoire	5.6
5.1.6 La structure d'actes de langage	5.9
5.1.7 Le type de relation reliant AGENT à ACTE INSTRUMENTAL	5.10
5.1.8 Définitions	5.11
5.2 RECONSTRUCTION DU NOUVEAU META-MODELE DE SAMPO ...	5.12
CONCLUSION	5.20

CHAPITRE 6 : VERS UNE NOUVELLE IMPLANTATION : EBAUCHE D'UNE SOLUTION

INTRODUCTION	6.1
6.1 QUELQUES CARACTERISTIQUES INDISPENSABLES DE LA NOUVELLE SOLUTION	6.2
6.2 DANS QUEL LANGAGE L'IMPLANTER ?	6.7

6.3 LE SCHEMA DE LA NOUVELLE BASE DE DONNEES	6.10
6.3.1 Transformation du schéma conceptuel en schéma MAG	6.10
6.3.2 Quelques requêtes intéressantes	6.14
6.3.3 Simplification du schéma MAG	6.17
6.3.4 Transformation du schéma MAG (après simplification)	6.24
CONCLUSION	6.25

CONCLUSION

BIBLIOGRAPHIE

INTRODUCTION

La modélisation est destinée à aider l'utilisateur à mieux se rendre compte des concepts manipulés par la méthode, de leurs propriétés ainsi que des relations entre eux, tandis que l'implantation vise, quant à elle, à réaliser un support informatique destiné à aider activement cet utilisateur dans sa tâche de spécification.

Ces deux réalisations détermineront le contenu du chapitre 4 de ce mémoire. Auparavant, les trois premiers chapitres auront respectivement pour but de présenter la méthode SAMPO, de comparer quelques modèles susceptibles de convenir pour modéliser cette méthode, et de donner un aperçu d'outils informatiques capables de faciliter son implantation.

Comme nous aurons l'occasion de nous en rendre compte, le chapitre 4 n'atteindra toutefois qu'en partie les objectifs qui lui étaient assignés et nécessitera, par conséquent, l'adjonction de deux chapitres supplémentaires.

Dans le chapitre 5, nous réaliserons un certain nombre d'aménagements de la méthode SAMPO et du modèle que nous en aurons élaboré, en vue de les rendre moins complexes et plus opérationnels. Et enfin, nous consacrerons le sixième et dernier chapitre de ce mémoire à la présentation de l'ébauche d'une nouvelle implantation possible.

CHAPITRE 1

PRESENTATION DE LA METHODE SAMPO

INTRODUCTION

Ce premier chapitre sera consacré à l'étude de la méthode SAMPO, l'acronyme de Speech Act based office Modeling aP-prOach.

Afin de replacer cette méthode dans son cadre théorique, la première section consistera en un rappel de la théorie des actes de langage. Cette section se basera sur les travaux effectués par [ROB89].

Dans la deuxième section, nous nous attacherons à définir l'approche SAMPO et son apport face à d'autres méthodes.

L'univers du discours fera l'objet de la troisième section : nous y verrons la manière dont SAMPO considère les différents éléments de cette partie du monde.

Enfin les concepts de l'analyse du discours, les propriétés des discours et les outils d'analyse du discours seront abordés dans la quatrième section.

1.1 LA THEORIE DES ACTES DE LANGAGE

Dans l'étude du langage, comme dans d'autres études systématiques, il n'y a pas de terminologie neutre. Cela conduit parfois à certains débats passionnés, à certaines "rivalités" entre différents auteurs.

1.1.1 Les débats et enjeux

1. Manfred Bierwisch [BIE80] reproche à beaucoup d'auteurs de ne pas faire la distinction entre le langage et la communication. L'un et l'autre doivent cependant, d'après lui, être distingués car il existe des cas où le langage est utilisé sans pour autant qu'il y ait communication (Ex: monologue) et, inversement, beaucoup de communications ne sont pas basées sur le langage (Ex: communications comportementales).
2. Dans la description du langage, outre la syntaxe et la sémantique, il est également question de pragmatique. Il est possible de distinguer au moins trois approches différentes face à cette pragmatique [SEA80] :
 - La première approche est de dire que la signification d'une expression est expliquée en terme des choses qu'elle désigne. On peut trouver une formulation récente de cette approche dans [STA72] :
"Syntax studies sentences, semantics studies propositions. Pragmatics is the study of linguistic acts and the contexts in which they are performed".
 - La deuxième approche est surtout l'oeuvre de Katz [KAT77]. La notion de base de la sémantique est le sens plus que la désignation. Le sens d'une expression, c'est-à-dire sa signification littérale, libre de tout contexte, doit être distinguée de sa signification réelle qui, elle, dépend du contexte. La sémantique, selon cette vue, étudie tous les aspects de la signification littérale des phrases et d'autres expressions, alors que la pragmatique s'attache aux conditions selon lesquelles les locuteurs et les auditeurs déterminent la signification des expressions d'après le contexte.

- La troisième approche, qui a pour maîtres Austin, Grice et Searle, a pour notion centrale l'utilisation des expressions d'un langage, cette utilisation étant expliquée en terme des intentions du locuteur. Contrairement à la deuxième approche, la signification littérale d'une expression, ici, ne sait pas être identifiée sans contexte. Austin, Grice et Searle ne voient pas de distinction claire entre sémantique et pragmatique, si ce n'est que la première est peut être une branche de la seconde.

Nous verrons que SAMPO se base plutôt sur cette troisième approche.

1.1.2 La notion d'acte de langage

La théorie des actes de langage part du principe que l'unité minimale de communication humaine n'est pas la phrase ou une autre expression, mais plutôt l'accomplissement de certains actes tels que faire des affirmations, poser des questions, donner des ordres, ... En exprimant une phrase, le locuteur exécute un ou plusieurs actes, mais l'acte en lui-même ne doit pas être confondu avec la phrase exprimée lors de son accomplissement.

Austin appelle ces actes des actes illocutoires. Ils sont souvent opposés à d'autres types d'actes comme les actes perlocutoires, qui sont les actes réalisés par l'acte de langage, ou les actes locutoires, qui sont l'acte de dire quelque chose au sens propre du terme.

Avant d'aborder les différents types d'actes, nous allons proposer un exemple qui illustrera la théorie exposée dans ce chapitre.

Il s'agit d'une conversation téléphonique entre un client (C) et la téléphoniste (T) d'une firme de vente par correspondance. Notons que cet exemple est totalement fictif.

- 1 T: Firme Petitpas, j'écoute !
- 2 C: Bonjour, combien coûte la chemise n°123 ?
- 3 T: Un moment, je vous prie...
- 4 *Quelques secondes plus tard...*

- 5 T: Allô ?
6 C: Oui.
7 T: La chemise n°123 coûte 1299 francs, cravate non comprise.
8 C: Merci,
9 J'en commande une.
10 T: Très bien,
11 Avez-vous un numéro de client ?
12 C: Non.
13 *Le client donne son nom et son adresse.*
14 *La téléphoniste les note.*
15 T: Vous avez le numéro de client 123456.
16 C: Oui... Quels sont les délais de livraison ?
17 T: Nous vous livrerons la chemise 123 endéans les trois semaines.
18 Nous vous donnerons également un cadeau de bienvenue.
19 C: C'est gentil !
20 P: Nous vous remercions de faire confiance à Petitpas pour vos achats.
21 C: Merci, au revoir.
22 T: Au revoir.

Exemple 1.1

1.1.3 Les types d'actes de langage

On peut distinguer trois types d'actes de langage proposés par Austin, bien que cette classification fasse l'objet de contestations.

L'acte locutoire : c'est l'acte de dire quelque chose au sens propre de terme ("act of saying"). L'acte locutoire exprime la signification de l'expression. Toutes les phrases de l'exemple 1.1 sont des actes locutoires.

L'acte illocutoire : c'est l'acte réalisé en disant quelque chose ("act performed in saying"). Par l'acte illocutoire le locuteur transmet la manière dont l'expression doit être comprise, c'est-à-dire la "force" de l'expression. Il y a donc toujours un acte illocutoire lorsqu'on dit quelque chose avec une intention.

Voici quelques verbes pouvant désigner des actes illocutoires [AUS62] : affirmer, décrire, avertir, remarquer, commenter, commander, ordonner, demander, critiquer, présenter des excuses, blâmer, approuver, souhaiter la bienvenue, promettre, reprocher, exiger et alléguer. Austin a affirmé qu'il y avait plus d'un millier d'expressions de ce type en anglais.

L'acte perlocutoire : c'est l'acte réalisé par l'acte de langage ("act performed by saying something"). Un acte perlocutoire est accompli par le locuteur si :

- (1) Il dit quelque chose (acte locutoire ou illocutoire)
- (2) Un effet est produit sur les sentiments, les pensées ou les actions de l'(des) auditeur(s).
- (3) Il existe un lien causal entre ces deux conditions appelées respectivement cause perlocutoire et effet perlocutoire.

Par exemple, si je soutiens un argument, je peux *persuader* ou *convaincre* mon interlocuteur ; si je l'avertis de quelque chose, je peux *l'effrayer* ou *l'inquiéter* ; si je lui demande quelque chose, je peux *l'amener à faire* ce que je lui demande ; si je lui fournis une information, je peux le *convaincre*, (*l'éclairer*, *l'édifier*, *l'inspirer*, *lui faire prendre conscience*) [SEA69]. Les expressions notées en italique ci-dessus désignent des actes perlocutoires.

Dans le cas de la conversation téléphonique (Exemple 1.1), en affirmant que la chemise 123 coûte 1299 francs, je peux *influencer* la décision de l'acheteur. La phrase 7 représente la cause perlocutoire tandis que la phrase 9 représente l'effet perlocutoire.

Tout acte locutoire est un acte illocutoire. En effet, on ne peut exprimer correctement la signification d'une expression (acte locutoire) sans exprimer la manière dont elle doit être comprise (force de l'acte illocutoire).

Par contre, on peut accomplir un acte illocutoire sans pour autant qu'il y ait un acte locutoire (Ex: geste). En d'autres termes, l'acte locutoire est une espèce de l'acte illocutoire dans le cas on l'on utilise le langage.

Par conséquent, nous ne parlerons pas d'actes locutoires.

Walter Cerf [CER69] remarque que la frontière entre acte illocutoire et acte perlocutoire est assez floue. L'acte illo-

cutoire étant l'acte *in saying* et l'acte perlocutoire étant l'acte *by saying*.

Ces quelques remarques montrent une certaine faiblesse dans la classification ; aussi faudra t-il être très prudent.

La littérature [AUR88] fait encore état de deux autres types d'actes de langage :

L'acte d'expression (utterance act) : c'est l'acte par lequel le locuteur émet des signaux phonétiques ou graphiques. Il s'agit donc de la forme donnée à l'expression (verbale, graphique, gestuelle, ...).

Dans le cas de la conversation téléphonique, l'expression sera évidemment toujours verbale.

L'acte propositionnel (propositional act) : c'est l'acte par lequel le locuteur exprime ce sur quoi porte l'expression. Il s'agit d'un acte de désignation et de prédiction. L'acte propositionnel fait partie de l'acte illocutoire.

La théorie des actes de langage de Austin et de Searle est à la base de la méthode SAMPO (Speech Act based office Modeling aPprOach) à laquelle nous allons désormais nous intéresser.

1.2 LA DEFINITION ET L'UTILITE DE SAMPO

a) Dans l'esprit de ses concepteurs, SAMPO se veut être, outre un outil d'analyse des communications entre personnes, une nouvelle approche de modélisation de l'information [LYY84a]. Cette modélisation serait basée sur la théorie des langages et sur une logique "intentionnelle", appelée logique illocutoire, afin de définir formellement la signification des actes de langage. Elle diffère donc, en ce sens, de la tradition en terme de modélisation de l'information. Selon cette approche, le système d'information est vu comme "un ensemble d'actes de langage pré-spécifiés, exécutables dans un contexte organisationnel ou social (l'univers des bureaux) donné, et formant un discours organisé et institutionnalisé".

Selon les concepteurs de SAMPO, les travailleurs de bureau s'engagent à des actions en faisant des promesses, posant des faits. A travers ces actions, ils créent, modifient, sup-

priment les engagements qui lient leur comportement actuel et futur. Le sens de l'information est déterminé par le changement potentiel qu'ils savent produire sur le monde social.

Pour SAMPO, un système d'information de bureau peut donc se définir comme "un discours structuré, dans un contexte organisationnel, qui consiste en un ensemble d'arrangements contractuels dans l'organisation, permettant à ses membres de contrôler, coordonner et établir des engagements".

b) La notion d'engagement est essentielle dans la théorie des actes de langage et par conséquent également dans SAMPO. En effet, chaque acte de langage a des conséquences impliquant d'autres actes et des engagements. D'autre part, une condition nécessaire au succès d'un acte illocutoire consiste en un engagement spécifique de celui qui exécute l'acte de langage. Dans la suite, nous entendrons par engagement un engagement à une action future.

Les sections suivantes seront consacrées à l'étude de l'univers du discours, i.e. la manière dont SAMPO voit les différents éléments de l'organisation, ainsi qu'à l'analyse du discours, i.e. la manière dont SAMPO peut délimiter et modéliser des discours complets et cohérents.

1.3 L'UNIVERS DU DISCOURS

L'univers du discours est formé du domaine entité et du domaine action [AUR88] [LEH86] [LYY84a].

1.3.1 Le domaine entité

Le domaine entité est un ensemble dont les éléments sont des entités statiques persistant dans le temps. Ces entités portent le nom d'entités du discours. Elles peuvent être des objets de transaction (articles, produits) ou des agents (fournisseurs, acheteurs). Elles sont souvent représentées par des noms dans le langage.

Le domaine entité est, en fait, un ensemble d'états qui se rattachent aux entités ainsi qu'un ensemble de lois ou de règles qui gouvernent leur comportement et leurs interactions.

1.3.2 Le domaine action

Le domaine action comprend des entités dynamiques qui portent le nom d'actes.

Ces actes sont dynamiques de deux manières : premièrement, ils introduisent des changements dans le domaine entité et, deuxièmement, ils sont momentanés et ont souvent une existence limitée, instantanée dans le temps. Dans certains cas cependant, ils peuvent s'étendre sur de plus longues périodes (activités de conception ou de construction).

Ces actes sont des éléments essentiels dans tout processus de transaction. Ils sont caractéristiques à l'organisation et indispensables à sa survie. Ils maintiennent, coordonnent l'accomplissement d'autres actes. Leur représentation dans le langage naturel prend souvent la forme de verbes (délivrer, ...).

Les actes sont des entités de second ordre, cela signifie que leur description implique nécessairement une référence aux entités du domaine entité. Par ailleurs, ils ont un initiateur, c'est-à-dire une personne ou un groupe de personnes qui accomplit l'acte.

Enfin, les actes peuvent être classés dans un des deux ensembles disjoints : l'ensemble des actes instrumentaux ou l'ensemble des actes de langage.

1.3.2.1 Les actes instrumentaux

Les actes instrumentaux sont des actions humaines qui peuvent accomplir des changements dans le domaine entité. Il existe toutefois des cas limites où l'implication humaine dans l'acte n'est pas immédiate comme, par exemple, l'inventaire automatique. Cependant, même dans ce cas, les actes exécutés ont finalement comme initiateurs des êtres humains.

Les états initiaux et les résultats des actes instrumentaux peuvent être décrits en se référant aux états d'entités du discours dans le domaine entité.

Si le résultat n'est pas établi, on dit que l'acte a échoué, or les échecs possibles des actes sont cruciaux pour le fonctionnement du système d'information dans son rôle de médiateur de transactions.

Les lois et réglementations sociales restreignent l'ensemble des actes instrumentaux possibles.

Les actes "livrer un article", "prendre note du nom et de l'adresse", ... sont des actes instrumentaux.

Ces actes peuvent encore être classés en processus, qui ont une persistance dans le temps (construire une maison), et en actes qui sont instantanés (livrer un article). L'initiateur d'un acte instrumental est un agent (la téléphoniste, dans le cas de la conversation téléphonique, est donc un agent). On peut également, dans le cas d'actes coordonnés et collectifs, identifier des ensembles d'agents.

Les actes instrumentaux dénotent quelque chose de fait, nous verrons que les actes de langage dénotent, quant à eux, quelque chose de dit.

1.3.2.2 Les actes de langage

Les actes de langage sont des actes instantanés qui ont comme résultat des expressions linguistiques ayant des significations définies dans un contexte organisationnel donné. Ainsi, par exemple, les actes "passer commande", "demander le prix", ... sont des actes de langage.

Les actes de langage comportent toujours au moins deux agents. Celui qui exécute un acte de langage est appelé locuteur, celui vers qui cet acte est dirigé est appelé auditeur.

Ces actes peuvent réussir ou échouer. Ils réussissent quand leur intention interne est reconnue par l'auditeur. Ils peuvent, par ailleurs, être défectifs lorsque les règles sous-jacentes sont violées lors de l'exécution de l'acte.

Les actes de langage ne produisent pas directement des changements dans le domaine entité sauf dans certains cas particuliers. Ils peuvent par contre "déclencher" des actes instrumentaux (la commande déclenchera l'acte instrumental de livraison).

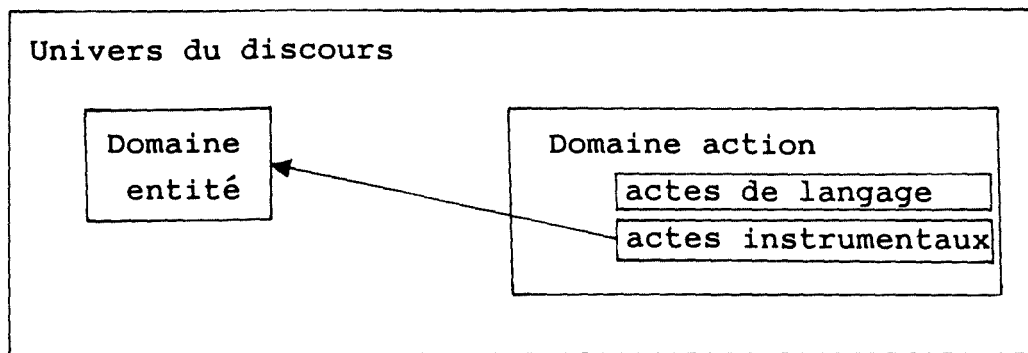


Figure 1.1

A) Les quatre aspects d'un acte de langage

Les actes de langage sont à la base de toute communication à l'intérieur d'une organisation et comprennent plusieurs catégories de "sous-actes" : les actes d'expression, les actes propositionnels, les actes illocutoires et les actes perlocutoires [AUR88].

Acte d'expression : c'est un acte qu'un locuteur réalise en émettant une expression comme, par exemple, "Allô" ou "J'en commande une".

Acte propositionnel : c'est un acte tel qu'en émettant le contenu propositionnel, le locuteur réalise un acte de désignation et de prédiction comme "Nous" et "livrerons la chemise 123 endéans les trois semaines" respectivement dans l'exemple 1.1.

Acte illocutoire : c'est l'unité de base d'une communication humaine signifiante. Il faut toujours que quelqu'un émette certaines expressions avec une intention. Ainsi, quelqu'un peut exécuter les expressions : "Je refuse de payer la note" ou encore, dans le cas de notre conversation, "Nous vous livrerons la chemise 123 endéans les trois semaines". Lorsque l'intention est reconnue par le(s) auditeur(s), l'acte illocutoire est réussi, et on dit que le sens de l'expression émise a été compris. Cela implique aussi qu'un engagement a été créé qui lie le comportement futur de toutes les parties et les engage dans certaines activités.

Acte perlocutoire : c'est un acte qui produit des effets sur les sentiments, l'attitude et le comportement futur des auditeurs. Ces effets sont qualifiés de perlocutoires puisque

causés par des actes perlocutoires. Un exemple d'effet perlocutoire est d'obtenir de quelqu'un, sur un ordre, qu'il livre la chemise 123 endéans les trois semaines.

En ce qui concerne les actes d'expression et les actes propositionnels, leurs définitions suffisent pour notre étude. Nous allons donc plutôt nous attarder sur les actes illocutoires et les actes perlocutoires, qui sont des notions fondamentales de la théorie linguistique de Searle et de Austin. Pour être plus exacts encore, la structure de ces deux types d'actes étant identiques, nous allons analyser en détail l'acte illocutoire.

B) L'acte illocutoire

Les actes illocutoires sont, soit élémentaires, soit complexes.

Un acte illocutoire complexe peut être construit récursivement sur un acte illocutoire élémentaire en utilisant des connecteurs illocutoires.

Si AIC est un acte illocutoire complexe, si AIE est un acte illocutoire élémentaire, et si P est une proposition, on peut écrire :

- (1) $AIC ::= AIE$
- (2) $AIC ::= \neg AIC$
- (3) $AIC ::= AIC \ \& \ AIC$
- (4) $AIC ::= P \implies AIC$

Soient a, a1 et a2 des actes illocutoires [LYY84a] :

Un acte illocutoire de la forme $\neg a$ est exécuté quand le locuteur, dans un certain contexte, fait savoir la non exécution de a. Par exemple : "Je ne promets pas de venir demain".

Un acte illocutoire de la forme $(a1 \ \& \ a2)$ est exécuté, dans un certain contexte, si les deux actes sont exécutés ensemble dans ce contexte. Par exemple : "La chemise n°123 coûte 1299 francs, cravate non comprise".

Un acte illocutoire de la forme $(P \implies a)$ est exécuté, dans un certain contexte, quand le locuteur, dans ce contexte, exécute un acte a si la condition P est vraie. En d'autres

termes, quand il exécute a , il exprime également que c'est fait à la condition P , et dans tous les contextes où P est vrai, a est également exécuté. Par exemple : "Si le niveau d'inventaire décroît en dessous du point de commande, je ferai une commande d'approvisionnement".

Les actes illocutoires exprimés par les règles (2) à (4) sont dits être sous forme canonique. En utilisant cette notation, nous pouvons définir la longueur d'un acte illocutoire. Celle-ci sera égale au nombre d'occurrences de connecteurs illocutoires $\neg$, $\&$, $\Rightarrow$ qui apparaîtront dans la notation canonique de a . En conséquence, la longueur d'un acte élémentaire sera 0.

C) Les composants de l'acte illocutoire

En général, un acte illocutoire élémentaire $F(P)$ a deux composants : une force illocutoire F et un contenu propositionnel P .

Plus formellement, la structure d'un acte illocutoire élémentaire peut cependant être présentée de la manière suivante [AUR88] :

Acte_ill (<contenu>, <contexte>, <force illocutoire>)

Tous ces constituants sont importants afin de comprendre la signification d'une expression.

1) Le contenu

Le terme "contenu" se réfère au contenu propositionnel du message, c'est-à-dire à ce sur quoi porte l'acte de langage. Le contenu propositionnel est toujours une partie de l'acte illocutoire. Il peut toutefois être absent dans des actes illocutoires tels que : Bravo, Aïe, Allô, ... [SEA69].

2) Le contexte

La même phrase dite dans différents contextes peut exprimer différentes forces illocutoires. Par exemple, la phrase 3 : "Un moment, je vous prie..." est une promesse de

la part de la téléphoniste de revenir bientôt ; dans une autre situation, on pourrait imaginer que cette même phrase intime l'ordre à l'employé d'attendre le bon gré de l'employeur. Le contexte est donc important, puisqu'il détermine la signification de la phrase. Il se définit en terme de locuteur, auditeur, place, temps, et monde possible :

Contexte (<locuteur>, <auditeur>, <place>, <temps>,
<monde possible>)

Le locuteur et l'auditeur : ils sont compris ici plutôt comme rôle organisationnel [LYY84a]. Dans le cas de la conversation téléphonique, le client et la téléphoniste sont, tour à tour, locuteur et auditeur.

La place : elle donne la coordonnée du locuteur dans l'espace, qui est utile pour la compréhension du sens de l'expression. Exemple : L'expression "Allô" n'a de sens que lorsque l'on (r)établit une conversation téléphonique.

Le temps : il identifie le monde réel d'expression sur l'axe du temps. Exemple : Une phrase du type "On va skier" exprimée en plein été sera comprise comme une blague alors qu'en hiver, elle n'aura pas ce sens.

Le monde possible : il inclut toutes les autres caractéristiques du locuteur, de l'auditeur, de la place et du temps qui sont nécessaires pour déterminer le contenu exact de la force illocutoire. Exemple : Selon que le locuteur est en position d'autorité ou non, la phrase 3 (Exemple 1.1) : "Un moment, je vous prie..." sera respectivement perçue comme un ordre ou une promesse.

3) La force illocutoire

La force illocutoire F est un composant nécessaire de tout acte de langage significatif. Chaque F détermine les relations sociales établies (les engagements) et la manière selon laquelle le contenu est révélé au monde. Ces relations sociales définissent les prévisions des comportements futurs des agents. Une force illocutoire possède 7 composants dont un est crucial : le point illocutoire ; viennent ensuite le mode de réalisation du point illocutoire, le degré de force du point

illocutoire, les conditions du contenu propositionnel, les conditions préparatoires, les conditions de sincérité, et enfin le degré de force des conditions de sincérité.

1° Le point illocutoire

Chaque type d'acte a un point ou une intention essentielle pour être un acte illocutoire. Par exemple, le point de cette assertion est d'expliquer comment les choses sont. Ce point ou intention est appelé point illocutoire.

Les points illocutoires peuvent être classifiés en 5 catégories :

1) Le point assertif ou descriptif ($\vdash$) :

Il décrit le monde. L'engagement concerne le locuteur.

Exemple :

"La chemise n°123 coûte 1299 francs".

2) Le point commissif ($\perp$) :

Il engage le locuteur à une action future.

Exemple :

"Nous vous livrerons la chemise n°123 endéans les trois semaines".

3) Le point directif (!) :

Il essaie d'obtenir de l'auditeur qu'il s'engage dans une action.

Exemple :

"J'en commande une".

4) Le point déclaratif ($\top$) :

Il décrit le monde et en le décrivant, il change ce monde afin de le faire correspondre au contenu propositionnel. L'engagement concerne le locuteur.

Exemple :

"Vous avez le numéro de client 123456".

5) Le point expressif (Υ , $\uparrow$) :

Il permet d'exprimer les états psychologiques (attitudes ou sentiments) du locuteur. L'engagement concerne le locuteur.

Υ exprime un désir et $\uparrow$ une opinion.

Exemple :

"Nous vous remercions de faire confiance à Petitpas pour vos achats".

Outre le point illocutoire, la force illocutoire comprend encore 6 autres composants :

2° Le mode de réalisation du point illocutoire

Généralement, un point illocutoire peut être réalisé de différentes manières [ROB89]. Par exemple, un point illocutoire directif peut être réalisé par un ordre ou par une demande. Dès lors, le mode de réalisation requiert que le locuteur fasse appel à sa position de pouvoir ou, au contraire, que la demande soit faite de manière plus humble.

3° Le degré de force du point illocutoire

Un point illocutoire peut être réalisé avec un degré de force plus ou moins grand. Par exemple, suggérer à quelqu'un de faire quelque chose est plus faible que l'y obliger ; le point illocutoire restant inchangé (directif).

4° Les conditions du contenu propositionnel

Le contenu propositionnel d'une force illocutoire ne peut être arbitraire mais doit satisfaire à certaines conditions appelées conditions du contenu propositionnel. Par exemple, le point illocutoire de l'expression "Dessinez un triangle carré" ne peut être réalisé, parce que le contenu propositionnel est impossible [LEH86]. Pour une promesse, le contenu propositionnel doit porter sur une action future du locuteur. Une expression comme "Je vous promets de livrer hier" n'a pas de sens [ROB89].

5° Les conditions préparatoires

Chaque acte illocutoire a des conditions préparatoires (pré-conditions) que le locuteur suppose vérifiées pour l'exécution de cet acte illocutoire dans un contexte donné.

Par exemple, une condition préparatoire pour une promesse est que le locuteur soit capable de la tenir. Pour une demande, une pré-condition est que l'auditeur ait des chances de pouvoir la satisfaire [ROB89].

6° Les conditions de sincérité

Il faut tenir compte, lors de l'exécution d'un acte illocutoire, des états psychologiques (désir, croyance, ...) relatifs au contenu propositionnel. Ces états sont appelés attitude propositionnelle. Par exemple, si le locuteur promet quelque chose à l'auditeur, il exprime son intention de faire ce qu'il promet. Dès lors, un locuteur est sincère s'il est dans l'état psychologique qu'il exprime, c'est-à-dire s'il a réellement l'intention de tenir sa promesse dans l'exemple ci-dessus.

7° Le degré de force des conditions de sincérité

L'état psychologique exprimé dans l'exécution d'un acte illocutoire peut être plus ou moins fort. Par exemple, un locuteur qui jure de faire quelque chose exprime une plus forte intention qu'un locuteur qui s'engage simplement à le faire.

Avant de passer à l'analyse du discours, il ne serait peut-être pas superflu de résumer les caractéristiques essentielles de SAMPO. Ainsi,

- Il existe deux domaines qui contiennent les entités et les actes.
- Les entités incluent des objets de transaction et des agents qui peuvent exécuter différents actes.
- Les actes, entités dynamiques, sont divisés en actes instrumentaux et en actes de langage.
- Les actes de langage comprennent quatre sous-actes : les actes d'expression, les actes propositionnels, les actes illocutoires et les actes perlocutoires.
- Les actes illocutoires sont élémentaires ou complexes.

- La signification d'un acte de langage peut être représentée par les actes illocutoires qui ont été exécutés.
- Les actes illocutoires ont une structure interne qui comprend la spécification de leur contenu propositionnel, de leur contexte, de leur force illocutoire et de leurs connecteurs illocutoires.
- La force illocutoire a sept composants dont un est crucial : le point illocutoire. Ce point peut être assertif, commissif, directif, déclaratif ou expressif.

Tel est l'univers du discours selon SAMPO ; voyons maintenant comment SAMPO analyse les discours.

1.4 L'ANALYSE DU DISCOURS

"Un discours est un effort soutenu de parole, c'est-à-dire une séquence de phrases individuelles" [AUR88]. En terme d'actes de langage, tant les actes de langage (complexes) individuels que les séquences d'actes de langage, c'est-à-dire les suites d'actes de langage, peuvent être étudiés comme des discours. [WUN80] prétend qu'aucun acte de langage n'est exécuté isolément. De plus, ces actes ne se suivent jamais selon une séquence arbitraire. Il dit, en effet, qu'une question appelle une réponse, qu'une proposition précède un accord, ...

Le but de l'étude du discours est de révéler des mécanismes qui permettent de garder le discours cohérent.

1.4.1 Les concepts de l'analyse du discours

L'analyse du discours utilisera les concepts suivants : le type de discours, la structure d'actes de langage, le segment de discours, le mouvement, le tour, la topicalisation [LYY84b] [AUR88] [ROB89].

1.4.1.1 Le type de discours

Un type de discours est la plus grande unité de communication qui puisse exister dans le système d'information de bu-

reau et visant à réaliser un but social, c'est-à-dire un but interne au type de discours.

Dans "Petitpas", par exemple, on peut identifier quatre buts sociaux :

Le premier est d'assurer le traitement des commandes clients reçues ainsi que leur livraison (initiale ou différée).

Le deuxième est relatif au traitement de tout ce qui concerne le paiement des clients.

Le troisième but, quant à lui, est de mettre à jour les stocks et de les réapprovisionner quand c'est nécessaire.

Enfin, on trouve encore un quatrième but qui est de mettre à jour les fichiers "clients" et "prospects" et de produire des statistiques de vente par produit et par client.

Les buts peuvent être clairs (envoyer des articles selon un critère spécifique de performance) ou ambigus (découvrir l'exactitude d'un discours scientifique).

1.4.1.2 La structure d'actes de langage

Une structure d'actes de langage est une séquence ordonnée d'actes de langage, généralement exécutés alternativement par les participants du discours, et qui forme un tout logique. Cette séquence possède donc des actes de langage initiaux et terminaux.

Cela signifie encore que si le locuteur prend la parole avec un acte appartenant à une certaine structure, l'auditeur est censé répondre par un acte appartenant à la même structure ; il peut toutefois interrompre cette dernière par une question [WUN80], comme dans la conversation téléphonique, pour demander quelques instants de patience.

Le plus souvent, les structures ne contiennent pas plus de deux ou trois actes de langage. Un exemple classique de structure est une question du locuteur suivie d'une réponse de l'auditeur. Exemple : "Quels sont les délais de livraison ?" ; "... trois semaines".

La notion de structure d'actes de langage est importante pour comprendre et analyser la structure d'un discours. Elle permet de démontrer qu'un discours est bien formé, c'est-à-dire cohérent et complet. Les deux aspects suivants montrent

également l'importance de la notion de structure d'actes de langage et de son étude :

- La signification de certains actes de langage ne peut être comprise que comme une partie d'une structure d'actes de langage et à un certain endroit dans cette structure. Par exemple, une réponse doit être donnée en réaction à une question et elle doit toujours suivre cette question.
- Généralement, dans une structure d'actes de langage, un acte de langage ne peut être suivi que par un ensemble spécifique d'alternatives, appelé "stage". Par exemple, un "stage" pour une question peut être une réponse, une demande d'explication, un refus ou une autre question. Exemple : La demande du prix est suivie d'une attente et ensuite de la réponse. L'ensemble des alternatives dépend du type de discours.

1.4.1.3 Les segments de discours

Les "stages", dans une structure d'actes de langage, peuvent être regroupés en un segment de discours. Les segments visent un sous-but utile pour atteindre le but social du type de discours concerné. Exemple : la demande du prix d'un produit est un sous-but ; ce dernier est indispensable au client pour pouvoir effectuer sa commande, qui est le but du discours.

On verra, par la suite, que la notion de segment est importante pour l'analyse de la cohérence et pour le contrôle du déroulement du discours [LYY84b].

1.4.1.4 Les mouvements

Les mouvements sont des actes de langage qui activent un stage dans un discours. Ils contrôlent le déroulement du discours. On distingue les mouvements initiaux, les mouvements de réaction et les mouvements de continuité, selon leur place dans la séquence.

Les mouvements initiaux : ils commencent le discours, c'est-à-dire déclenchent un acte de langage spécifique appartenant à un stage initial dans un discours. Souvent, les ques-

tions et les demandes sont des mouvements initiaux [WUN80] (Ex: la demande de prix).

Les mouvements de réaction : ils ont pour but de réagir au déroulement du discours ou de l'affecter. On peut distinguer les mouvements d'acceptation et les mouvements de rejet. On peut encore parler de mouvements de réaction neutres servant à quitter la question laissée ouverte. Ces mouvements clôturent simplement les segments de discours et assurent la continuité du discours. Une confirmation, un rejet ou une réponse constituent souvent des mouvements de réaction [WUN80] (Ex: le prix est de X).

Les mouvements de continuité : ils n'affectent pas directement le développement du discours mais assurent simplement la continuité. Pour ceux-ci, on ne dispose pas de caractéristiques précises afin de les identifier (Ex: Un moment, je vous prie ...).

1.4.1.5 Le tour

La prise de tour définit la manière dont les différents participants peuvent apporter leur contribution au discours [LYY84b], c'est-à-dire comment (où et quand) l'un des participants peut entrer dans le discours et exécuter un acte de langage. L'exemple de la conversation téléphonique montre nettement la prise de tour aux différents endroits du discours.

1.4.1.6 La topicalisation

La topicalisation consiste à définir les aspects du monde, dans le contexte d'expression, qui sont mis en évidence dans un stage d'un segment de discours. Les segments partagent une même topicalisation. Exemple : Le segment concernant la demande du prix d'un produit aura comme topicalisation cette dernière.

Elle est exprimée par la forme canonique d'un acte illocutoire $F(P)$; par exemple : !(prix de la chemise 123), pour la topicalisation du segment concernant le prix de la chemise 123. La force illocutoire de l'acte exprime ici quels engagements l'acte a activé. Le contenu propositionnel, pour sa

part, révèle les objets et leurs propriétés qui concernent ce stage. Les objets peuvent être des entités ou des actes.

La topicalisation peut être changée en modifiant la force illocutoire, le contenu propositionnel ou les deux. La modification du contenu propositionnel introduit de nouveaux segments de discours, tandis que la modification de la force illocutoire décrit une évolution, une création, un échange d'engagements à l'intérieur d'un même segment de discours.

1.4.2 Les propriétés des discours

Le but de l'analyse du discours est de démontrer que le discours étudié est bien formé, c'est-à-dire cohérent et complet. La cohérence définit dans quels contextes un acte illocutoire peut être exécuté avec succès, et la complétude détermine le plus grand ensemble possible d'actes illocutoires qui réussissent dans un contexte donné.

Nous allons exposer brièvement la cohérence et la complétude, le lecteur intéressé par plus de détails sur ce sujet pourra consulter la littérature [ROB89] [AUR88] [LYY84b].

1.4.2.1 La cohérence

L'étude de la cohérence vise à montrer comment les expressions d'un discours sont logiquement reliées entre elles. Par exemple, on peut considérer qu'une séquence formée d'une question suivie d'une réponse est cohérente [ROB89].

Un discours cohérent n'inclut que des actes illocutoires qui réussissent, c'est-à-dire qui réalisent leur but dans un contexte d'expression donné.

Pour étudier la cohérence, on étudie la "stage structure" d'un discours. Pour chaque stage, on définit les conditions nécessaires et suffisantes pour que chaque acte illocutoire alternatif existant réalise son but. Par exemple, à la réception d'une commande d'un client, il faut spécifier sous quelles conditions la commande peut être acceptée, refusée, ...

1.4.2.2 La complétude

La complétude d'un discours signifie, dans un certain sens, que ce discours est complètement spécifié, c'est-à-dire qu'il contient tous les actes de langage exécutables. On peut définir deux notions de complétude : la complétude faible et la complétude forte.

Un discours est faiblement complet : si et seulement s'il est cohérent et si toutes les structures d'actes de langage qu'il contient se terminent. Cela signifie que le discours se termine dans tous les cas et donc qu'il ne peut y rester des engagements en suspens.

Un discours est fortement complet : si et seulement s'il est faiblement complet et s'il contient tous les actes de langage réalisables et leurs structures possibles. Cette définition implique que dans un "stage" donné d'un discours, toutes les alternatives possibles apparaissent dans la spécification de ce discours. Généralement, l'énumération exhaustive de toutes ces alternatives pour chaque "stage" du discours est impossible. De plus, la théorie des actes de langage ne fournit présentement aucun ensemble de critères qui permettrait de juger de la complétude d'un discours. L'analyse du discours veillera donc à ce qu'un discours soit le plus complet possible.

Il faut encore noter qu'un discours peut être ambigu et ce, principalement, pour deux raisons :

- (1) Certaines activités de bureau ne peuvent être exprimées précisément au moyen du langage ; on se réfère trop à des connaissances tacites partagées par différents acteurs.
- (2) La nature politique de beaucoup d'activités tire profit d'une ambiguïté élevée.

1.4.3 Les outils d'analyse du discours

SAMPO offre deux modèles de description graphique, complétés par plusieurs tables, pour décrire les différents éléments d'un discours de manière détaillée. Ces outils aident à analyser les problèmes de communication existants dans ce dis-

cours et à assurer sa qualité, c'est-à-dire sa cohérence et sa complétude [LYY84b] [AUR88] [ROB89] ;

- (1) En trouvant les incohérences du discours.
- (2) En contrôlant de manière plus efficace les engagements.
- (3) En pouvant réorganiser les processus de discours existants et en créer de nouveaux.

Les outils graphiques sont, d'une part, le graphe de discours et, d'autre part, le graphe de conversation. Ces deux graphes sont des réseaux cycliques d'actes. Leurs noeuds consistent en des ensembles d'éléments simples. "Un graphe de discours correspond à un type de discours dans lequel chaque séquence d'actes de langage déclenchée donne lieu à un graphe de conversation" [ROB89].

1.4.3.1 Les graphes de discours

Un graphe de discours décrit la structure générale d'un discours, c'est-à-dire les possibilités de conversation entre les différents participants au discours. Il constitue un modèle de la structure du discours institutionnalisé et organisé.

Un graphe de discours est représenté par les symboles de la figure 1.2. Les éléments sont [LYY84b] [AUR88] [ROB89] :

- un ensemble d'activités
- un ensemble d'actes (de langage ou instrumentaux) du domaine action
- un ensemble d'entités du domaine entité
- un ensemble de prédicats
- un ensemble de participants au discours
- un ensemble d'actes illocutoires (élémentaires ou complexes)
- un ensemble de relations entre ces objets.

"Une activité est une routine d'action institutionnalisée régie par une série d'arrangements organisationnels (contrats). La notion d'activité permet d'assigner des actes instrumentaux et des actes de langage à des fonctions organisationnelles distinctes. Ceux-ci définissent des processus de transaction et de négociation pour les différents membres de

l'organisation" [AUR88]. Ces activités, à leur tour, sont formées par des positions, qui définissent les responsabilités des membres pour les différentes phases des processus de transaction. Les gens occupant des positions sont donc responsables, par exemple, pour les phases de transaction de facturation, d'entrée de commande, ...

Les activités peuvent être réparties, selon la nature des actes, dans trois ensembles disjoints :

- (1) Les activités d'actes de langage : dans lesquelles les engagements sont créés, contrôlés et coordonnés.
- (2) Les activités d'actes instrumentaux : dans lesquelles les agents n'exécutent que quelques processus d'échange.
- (3) Les activités combinées : dans lesquelles il y a négociation et les engagements sont honorés.

Parmi les prédicats utilisés dans un graphe de discours, on distingue :

- les prédicats de sélection (selection predicates) qui conduisent la sélection d'actes parmi plusieurs alternatives.
- les prédicats de porte (gate predicates) qui permettent l'exécution d'actes uniquement quand certaines conditions sont vérifiées.
- les prédicats de déclenchement (triggering predicates) qui déterminent quand une conversation est déclenchée.

Les prédicats de déclenchement contrôlent l'initialisation d'un discours, les prédicats de sélection et de porte, par contre, concernent son évolution.

La topicalisation est représentée à deux niveaux dans un graphe de discours.

Au niveau de base, la topicalisation concerne les entités et les actes instrumentaux.

Au niveau secondaire, la topicalisation concerne les actes de langage eux-mêmes.

En effet [ROB89], "il y a, dans les systèmes d'information, des actes de langage qui peuvent être dérivés d'autres actes de langage. Par exemple, les statistiques mensuelles des ventes sont basées sur différents actes de langage concernant chaque vente réelle". Ce type de situation est ap-

pelé un engagement illocutoire faible (weak illocutionary commitment), c'est-à-dire un engagement créé par un acte de langage virtuel (acte de langage pour lequel le locuteur ou l'auditeur n'est pas humain comme, par exemple, dans le cas d'outputs automatiques du système d'information).

Nous pouvons encore identifier la limite d'un discours. Au delà de cette limite, les actes de langage ou actes instrumentaux sont qualifiés d'externes. Ce sont des actes dont l'auditeur, le locuteur ou l'agent (pour les actes instrumentaux) n'est responsable d'aucune activité du type de discours.

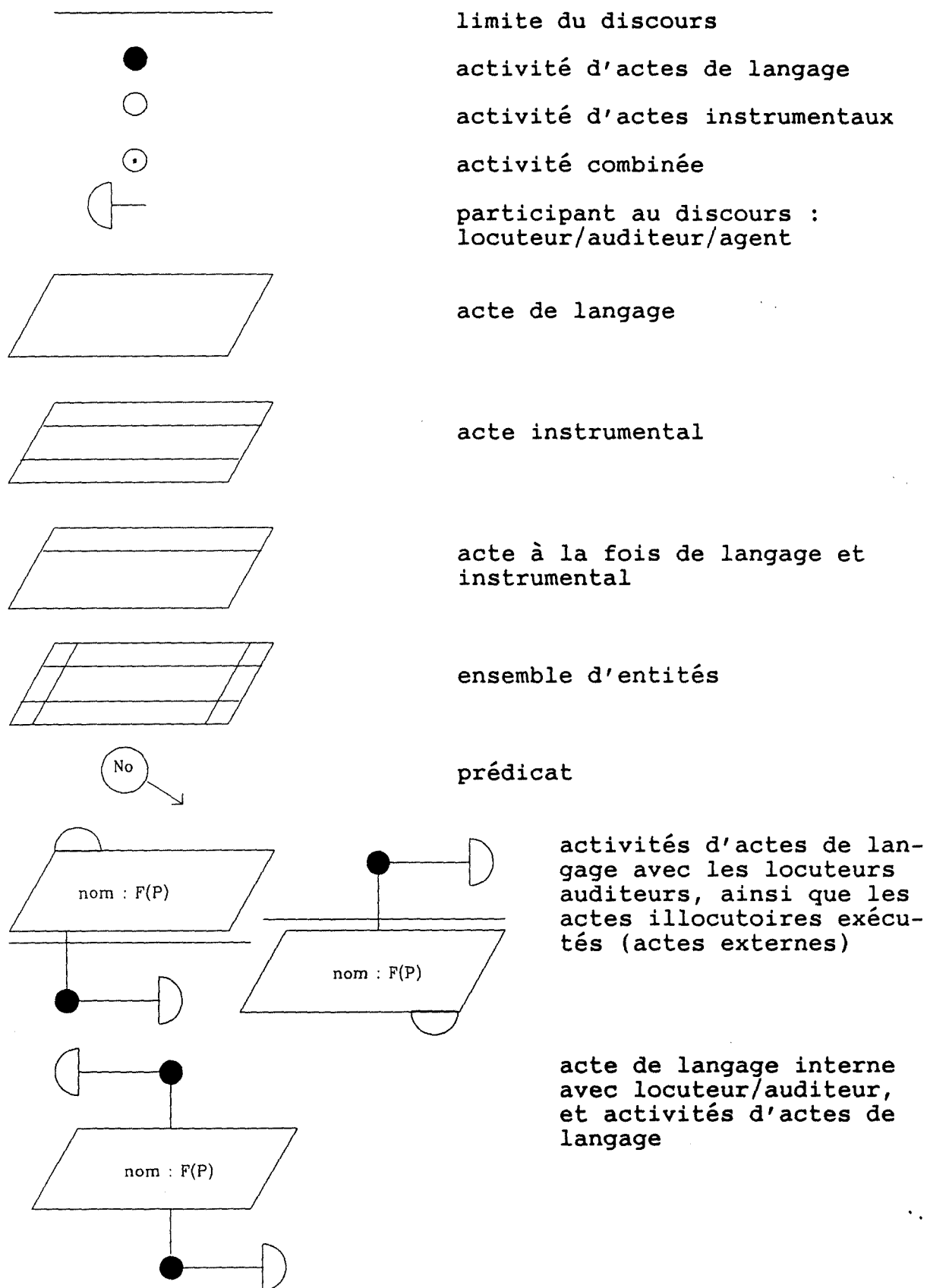


Figure 1.2. Symboles utilisés dans les graphes de discours

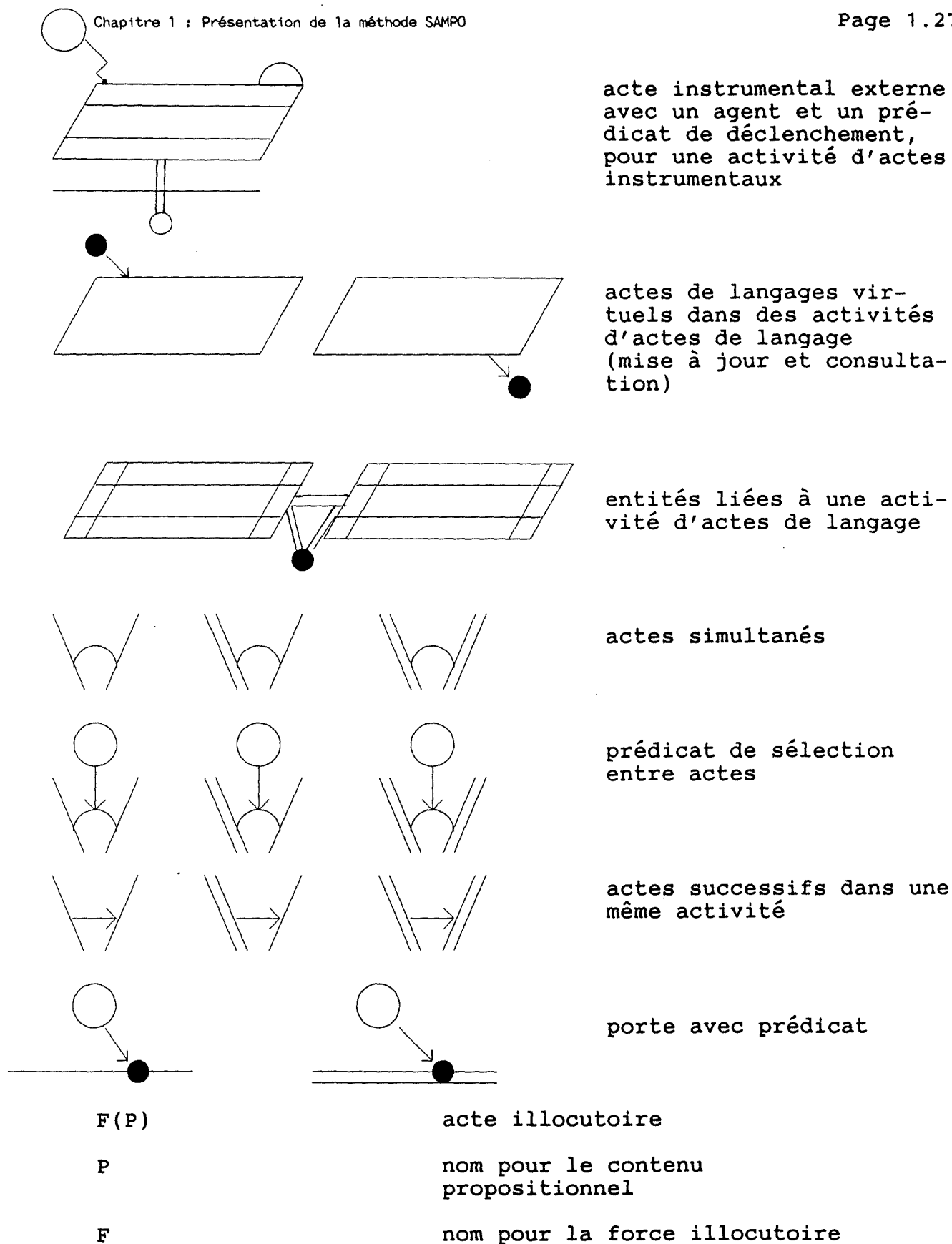


Figure 1.2. Symboles utilisés dans les graphes de discours
(suite)

POINTS ILLOCUTOIRES :

┐	point illocutoire assertif
!	point illocutoire directif
└	point illocutoire commissif
┐	point illocutoire déclaratif
Ψ, ↯	points illocutoires expressifs
!	point illocutoire de requête
-	(directif)

CONNECTEURS ILLOCUTOIRES :

&	conjonction illocutoire
┐	négation illocutoire
==>	implication illocutoire

Figure 1.2. Symboles utilisés dans les graphes de discours
(fin)

1.4.3.2 Les graphes de conversation

Un graphe de conversation décrit les structures d'actes de langage, les actes instrumentaux et leurs dépendances dynamiques. On construit un graphe de conversation pour chaque ensemble d'actes de langage déclenché dans le graphe de discours.

Un graphe de conversation est représenté par les symboles de la figure 1.3. Les éléments sont [LYY84b] [AUR88] [ROB89] :

- un ensemble d'actes de langage
- un ensemble d'actes instrumentaux
- un ensemble de locuteurs
- un ensemble d'auditeurs
- un ensemble d'actes illocutoires
- un ensemble comprenant les marques de début et de fin
- un ensemble de prédicats.

Les graphes de conversation contiennent aussi des propriétés qui décrivent la prise de tour de parole ("turn-taking"), la topicalisation du discours, ainsi que les contraintes structurelles pour l'exécution simultanée et successive d'actes [ROB89].

1.4.3.3 Les tables

SAMPO offre encore plusieurs outils tabulaires décrivant de manière plus détaillée les différents éléments du discours repris dans les graphes de discours et de conversation afin de les compléter.

Voici les différentes tables que SAMPO met à notre disposition :

- la table des types de discours
- la table des actes de langage
- la table des actes instrumentaux
- la table des entités
- la table des activités
- la table des positions
- la table des prédicats de déclenchement
- la table des prédicats de sélection
- la table des prédicats de porte

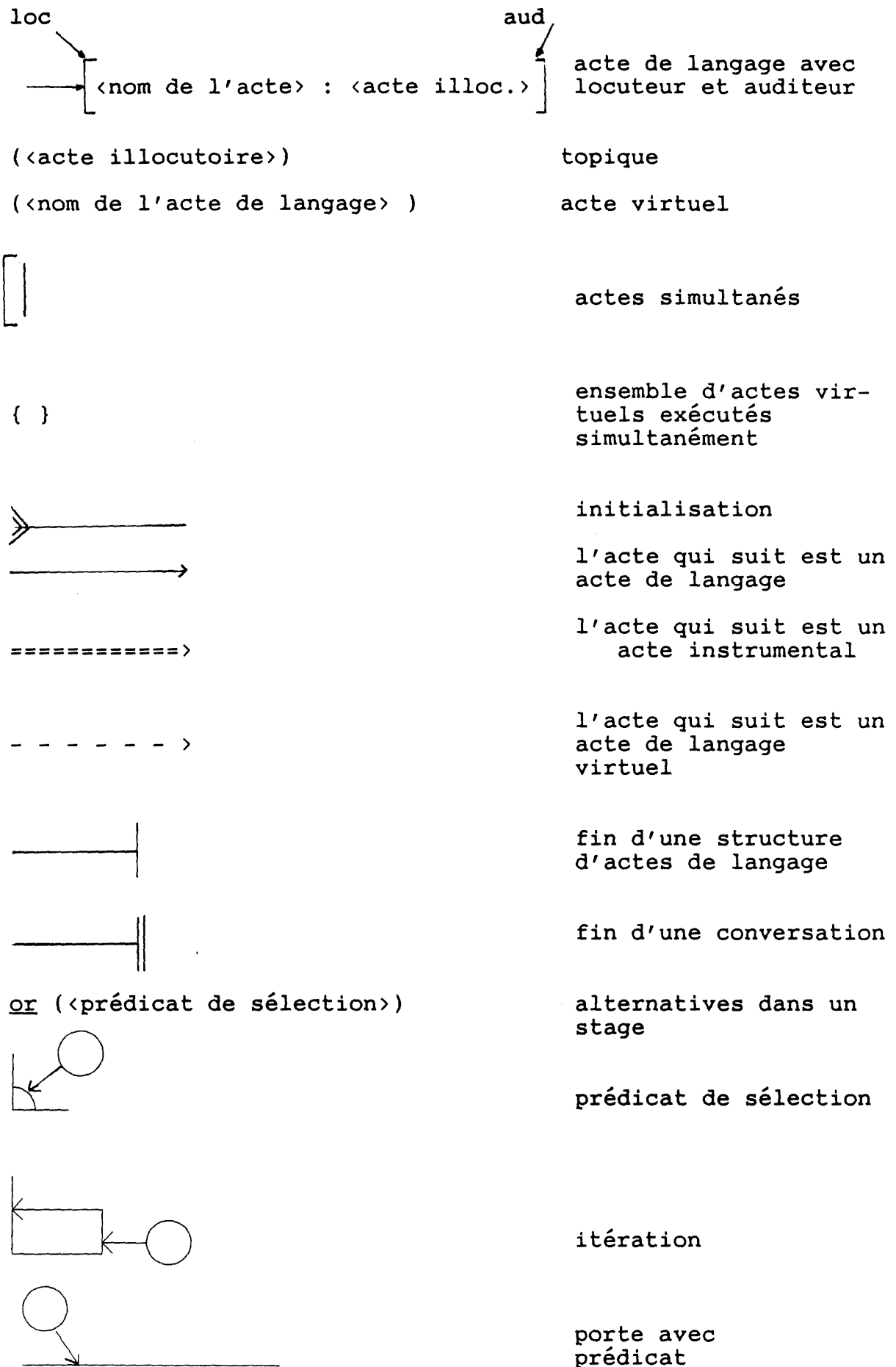


Figure 1.3. Symboles utilisés dans les graphes de conversation

CONCLUSION

Ce premier chapitre était consacré à la présentation de la méthode SAMPO (Speech Act based office Modeling aPprOach), et plus spécialement de sa partie permettant d'analyser un discours organisationnel.

Nous avons vu que nous avions affaire ici à une méthode d'analyse des communications permettant de spécifier des systèmes d'information en appréhendant la nature sociale des activités de bureaux ; cette méthode n'ayant de fondements que par la théorie des langages.

L'univers du discours et l'analyse du discours de SAMPO sont, en effet, entièrement basés sur cette théorie.

Le chapitre suivant, quant à lui, aura pour but d'étudier et de comparer quelques modèles de structuration des informations susceptibles de permettre la modélisation de la méthode SAMPO ou, plus particulièrement, la modélisation de l'univers du discours et des concepts de l'analyse du discours.

CHAPITRE 2

COMPARAISON DE QUELQUES MODELES DE STRUCTURATION DES INFORMATIONS

INTRODUCTION

Afin de modéliser la méthode SAMPO, c'est-à-dire de représenter les différents concepts qu'elle fait intervenir ainsi que les relations entre eux, on peut avoir recours à plusieurs modèles de structuration des informations.

Dans un premier temps, nous étudierons brièvement ces modèles en évoquant leurs caractéristiques respectives.

Ensuite, à l'aide d'un exemple, nous essayerons de choisir le meilleur d'entre eux.

Enfin, nous aborderons la notion de méta-modèle qui sera une des notions centrales des chapitres suivants.

2.1 LES MODELES

Pour représenter les différents concepts de la méthode SAMPO, il est nécessaire, voire indispensable, d'introduire ici la notion de modèle de structuration des informations.

Un modèle de ce type "sert à définir la sémantique des données qui appartiennent à la mémoire (ou base) des informations du S.I. ou qui sont véhiculées par des messages. La structuration des informations comporte notamment la définition des données et relations entre celles-ci. Précisons que la mémoire du S.I. comprend toutes les informations stockables sur des supports informatiques ou manuels" [BOD89].

Il existe de nombreux modèles conceptuels de structuration des informations.

2.1.1 Le modèle Objet

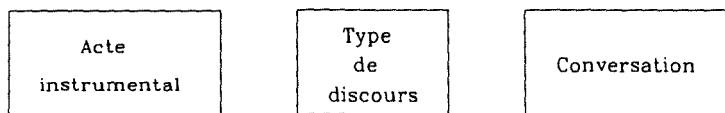
C'est le modèle le plus simple. Il n'offre de représentation que pour les objets. Un objet se définit comme suit [WEL88a] : "Une chose qui existe par elle-même et est représentée par les propriétés qui lui sont associées". Un objet est à rapprocher d'une entité qui consiste en : "Une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations. Une entité n'existe en tant que telle que par rapport à un individu ou un groupe qui la considère comme un tout, lui confère une existence autonome et la distingue d'autres entités et de son environnement (...)" [BOD89].

On remarque qu'un objet correspond souvent, au niveau sémantique, à un nom [WEL88a]. Exemples : un client, une commande, ... ou encore, dans le cas qui nous préoccupe, un acte instrumental, un type de discours, une conversation, ...

La classe de tous les objets possibles du réel perçu qui vérifient la définition constitutive du type porte le nom de type d'objet.

Un type d'objet est représenté par un rectangle.

Exemples :



2.1.2 Le modèle Binaire

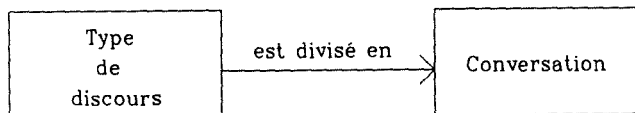
Ce modèle est plus élaboré, en ce sens que deux objets peuvent maintenant être mis en relation ; une relation étant une association entre ces objets [WEL88a].

Un verbe peut représenter une relation au niveau sémantique. Exemples : un client **passé** une commande, un type de discours **est divisé en** conversation(s).

La classe de toutes les relations possibles du réel perçu qui vérifient la définition constitutive du type porte le nom de type de relation.

Dans le modèle binaire, un type de relation est représenté par une flèche. Le nom du type de relation est attaché à la flèche.

Exemple :



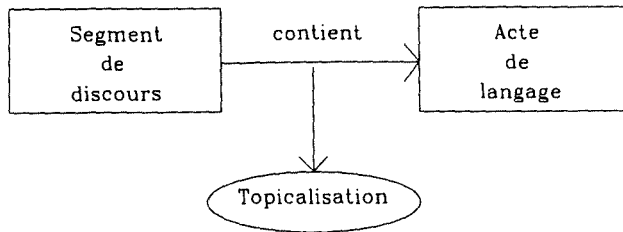
On distingue deux types de modèles binaires :

Le modèle Binaire-1, qui n'accepte pas de propriétés de relation et le modèle Binaire-2, qui accepte des propriétés de relation.

Une propriété ou un attribut est une caractéristique associée aux objets, ou aux relations [WEL88a]. Elle est difficilement repérable sémantiquement car elle peut prendre la forme d'un adjectif, d'un adverbe, ...

Dans le modèle binaire, les propriétés de relation sont représentées par un ovale relié à la flèche du type de relation.

Exemple :



2.1.3 Le modèle Entité-Relation (E-R)

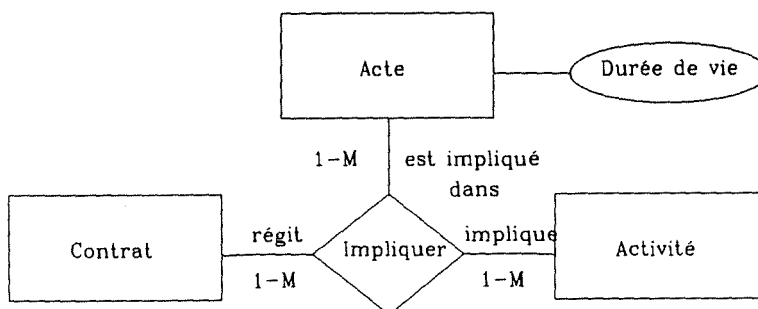
Ce modèle, tout en reprenant les caractéristiques du modèle binaire, en ajoute une importante : la relation peut être multiple (binaire, ternaire, quaternaire, ...), c'est-à-dire impliquer un, deux ou plusieurs objets, non nécessairement distincts.

Chaque objet, dans une relation de ce type, assume un rôle donné. Le couple formé par le nombre minimum de fois et le nombre maximum de fois qu'un rôle doit ou peut (respectivement) être joué, à tout moment, par l'ensemble des objets d'un certain type, porte le nom de "contrainte de connectivité". Lorsque le nombre maximum de fois peut être aussi élevé que l'on veut, l'initiale M est utilisée pour signifier "Many".

Graphiquement, les noms de rôles et les couples indiquant les connectivités sont attachés aux liens qui séparent le type de relation des types d'objet sur lesquels il porte.

Le type de relation, pour sa part, est désormais représenté par un losange.

Exemple :



On relève par ailleurs, dans ce modèle, une classification identique à celle présente dans le cas du modèle binaire.

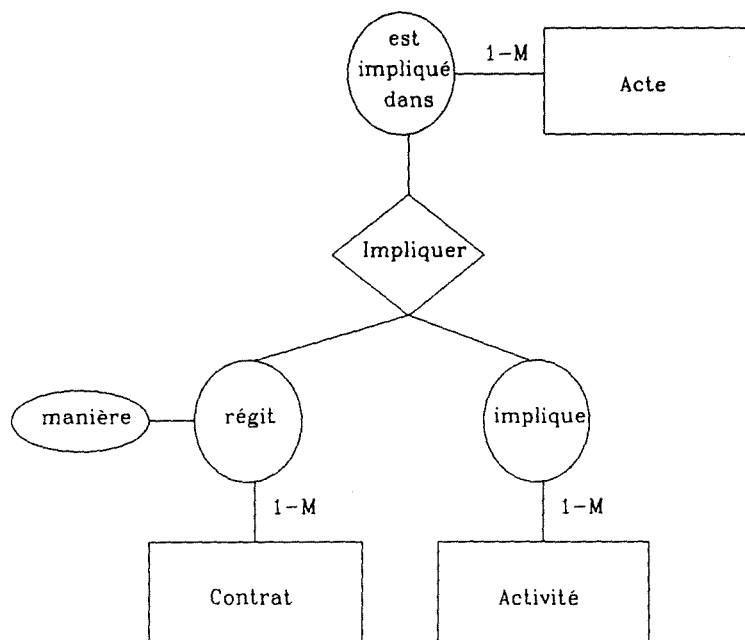
On peut distinguer, en effet, le modèle Entité-Attribut-Relation (EAR) qui correspond au modèle Binaire-1 puisqu'il n'admet pas de propriétés de relation et le modèle Entité-Relation-Attribut (ERA) qui correspond, lui au modèle Binaire-2 puisqu'il accepte de telles propriétés.

Notons encore que le modèle ERA, issu des travaux de P. Chen [CHE76], est très répandu et qu'il a fait, et fait encore, l'objet de toute une série d'extensions comme, par exemple, celles qui consistent à lui ajouter des concepts tels que la généralisation ou le regroupement. Ces extensions ont donné lieu à de nouveaux modèles tel le modèle Entité/Association (E/A) décrit dans [BOD89] et qui inclut, notamment, d'autres contraintes d'intégrité que les contraintes de connectivité.

2.1.4 Le modèle Objet-Propriété-Rôle-Relation (OPRR)

Pour terminer cette revue des modèles, nous pouvons encore évoquer un modèle récent : le modèle OPRR. Il ajoute au modèle E-R la possibilité de spécifier des propriétés pour les rôles, c'est-à-dire des caractéristiques associées aux rôles. Graphiquement, ces propriétés sont toujours représentées par des ovales mais les rôles, en ce qui les concerne, sont désormais placés dans des cercles.

Exemple :



Dans le tableau 2.1, nous avons résumé les caractéristiques principales des différents modèles évoqués.

Modèle	Description
Objet	Seuls les objets peuvent être représentés
Binaire :	Deux objets peuvent être associés par une relation
Binaire-1	Relation sans propriétés
Binaire-2	Relation avec propriété(s)
Entité-Relation (E-R)	Plusieurs objets peuvent participer à une relation
EAR	Relation sans propriétés
ERA	Relation avec propriété(s)
Objet-Propriété-Rôle-Relation (OPRR)	Plusieurs objets dans une relation, chacun participant selon un rôle explicite. Les relations et les rôles ont des propriétés.

Tableau 2.1

2.2 LE CHOIX D'UN MODELE

2.2.1 Les critères de choix

Comment choisir un bon modèle si ce n'est en énonçant une liste de critères permettant d'évaluer chacun d'entre eux.

On pourrait dès lors être tenté de dresser une grille comparative de tous ces modèles avec, en ordonnée, le nom de

ceux-ci et, en abscisse, une série de critères comme : le modèle est-il lisible ? complexe ? n'y a-t-il pas de perte d'informations ? tout est-il représentable avec ce modèle ? prend-il beaucoup de place ? est-il relativement facilement compréhensible pour un néophyte ?

On pourrait même attribuer un poids aux critères de sorte que le modèle cherché serait celui dont le poids est maximum.

Cela n'est, hélas, pas faisable ainsi. Car, comme nous allons le voir dans la section suivante, le modèle adéquat dépend beaucoup de ce qu'on doit modéliser.

2.2.2 Le choix

Afin d'essayer de choisir un modèle, nous allons prendre plusieurs exemples qui correspondront à des niveaux de difficulté croissants dans la spécification. Chaque exemple sera présenté au moyen des différents modèles vus précédemment. Le modèle objet sera cependant ignoré car, dès le premier niveau, il se révélera insuffisant étant donné la nécessité de pouvoir spécifier des relations entre objets.

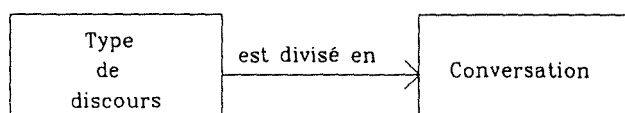
2.2.2.1 Niveau 1 :

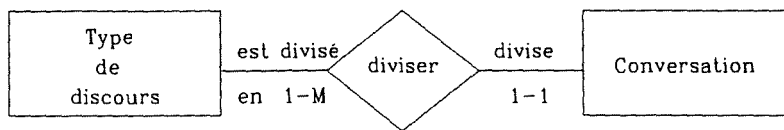
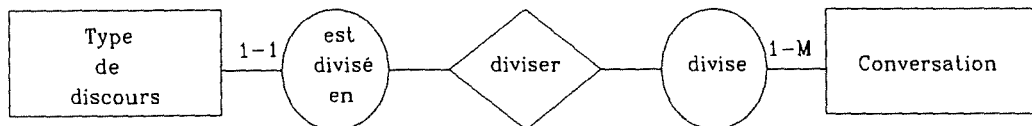
Exemple : Le type de discours est divisé en une (ou plusieurs) conversation(s).

Représentations :

La représentation de cet exemple au moyen des différents modèles donnerait :

Modèle Binaire (1 & 2)



Modèle E-R (EAR & ERA)**Modèle OPRR****Discussion**

La première chose à mentionner ici est que, comme on peut s'en rendre compte, les contraintes de connectivité ne sont pas placées au même endroit dans le modèle E/R et dans le modèle OPRR. Bien que définies de la même manière, elles sont en effet, inversées d'un modèle à l'autre et ce sans raison particulière. Il est toutefois nécessaire de le savoir pour pouvoir interpréter correctement les schémas. Cela étant dit, on peut également remarquer que, dans cet exemple, tous les modèles considérés ont été capables d'exprimer la spécification proposée. Au vu des résultats, le modèle binaire semblerait cependant plus adéquat que les autres, car il n'est pas encombré d'éléments compliquant la représentation du modèle.

Par conséquent, si ce niveau était le niveau de spécification le plus complexe rencontré, nous pourrions conclure que c'est le modèle Binaire-1 qui convient le mieux, car il est efficace sans être compliqué.

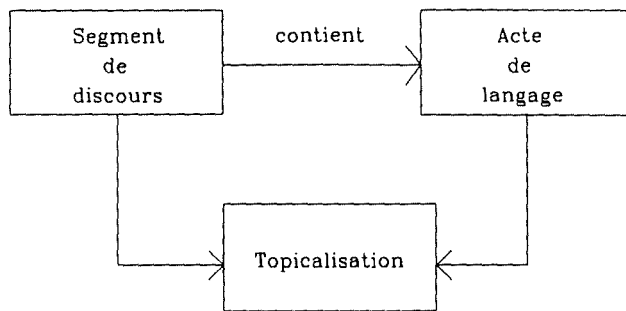
2.2.2.2 Niveau 2 :

Exemple : Un segment de discours (cf. 1.4.1.3) contient des actes de langages ; la relation est caractérisée par une topicalisation (cf. 1.4.1.6).

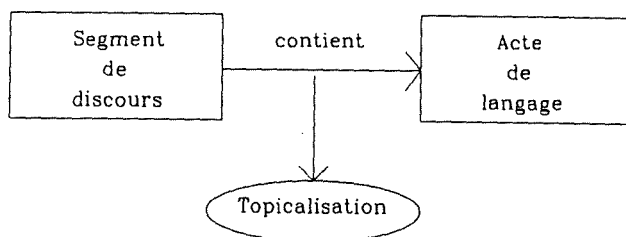
Représentation :

La représentation de cet exemple au moyen des différents modèles donnerait :

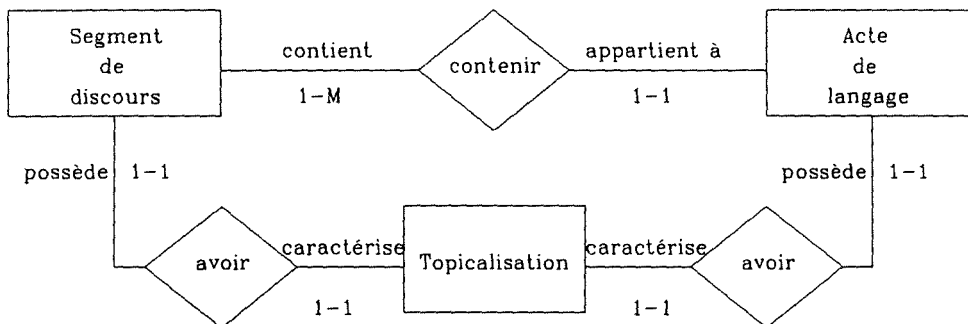
Modèle Binaire-1



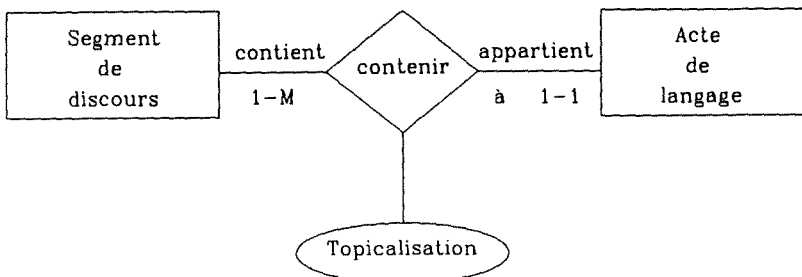
Modèle Binaire-2

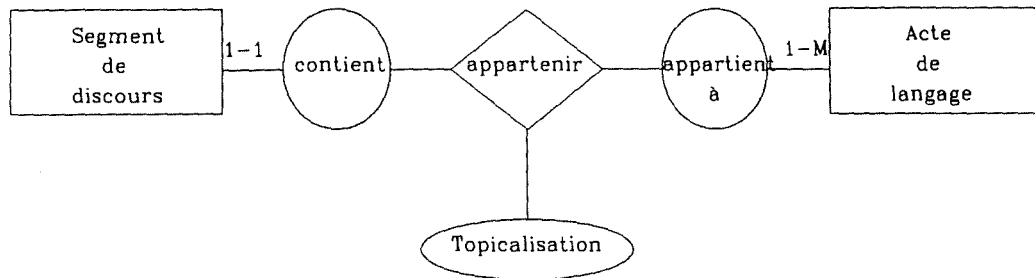


Modèle EAR



Modèle ERA



Modèle OPRR**Discussion**

Si nous regardons le modèle Binaire-1 (et le modèle EAR), nous observons qu'il est nécessaire, à présent, d'introduire un type d'objet supplémentaire pour pouvoir représenter la propriété de la relation. Il n'y a donc pas de perte d'informations, mais ces modèles donnent lieu à une manière de représenter l'information non naturelle et propice aux erreurs. Le nouveau type d'objet n'existe en effet que par la relation "contenir".

Les modèles Binaire-2 et ERA par contre conviennent mieux dans ce cas. Pour eux, la relation "contenir" reste une relation simple et une propriété, "Topicalisation", la qualifie.

Pour cet exemple, nous pouvons donc conclure que les modèles ERA et Binaire-2 sont les plus adéquats.

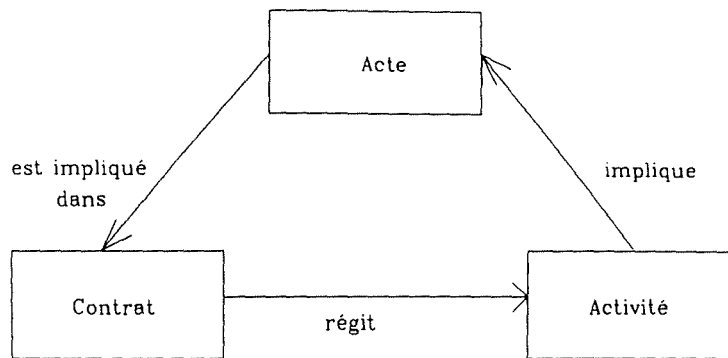
2.2.2.3 Niveau 3 :

Exemple : Une activité est une routine d'action institutionnalisée régie par une série d'arrangements organisationnels (contrats). La notion d'activité permet d'assigner des actes instrumentaux et des actes de langage à des fonctions organisationnelles distinctes (cf. 1.4.3.1).

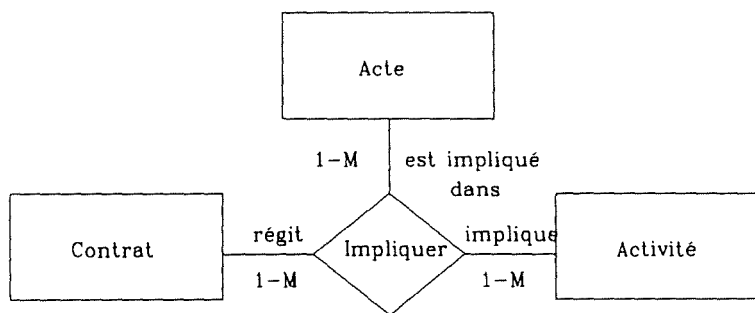
Représentation :

Les différents modèles de données donneraient :

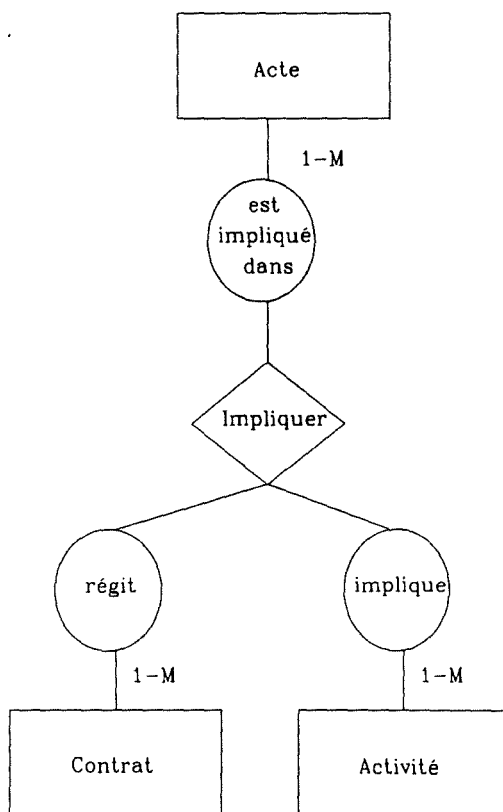
Modèle Binaire (1 & 2)



Modèle E-R (EAR & ERA)



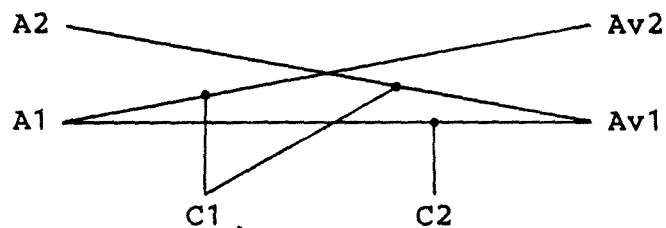
Modèle OPRR



Discussion

En examinant les différents résultats obtenus, il apparaît que nous avons réussi, au prix d'une certaine complexité, à modéliser l'information avec chaque modèle. Pour ce qui est des deux modèles binaires cependant, nous devons constater une perte d'informations.

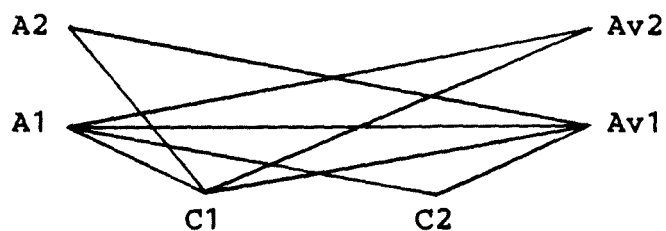
En effet, considérons la représentation du problème avec le modèle ERA (ou EAR). Si on dresse un schéma de quelques occurrences, on peut avoir :



ou encore :

Impliquer		
Acte	Activité	Contrat
A1	Av1	C2
A1	Av2	C1
A2	Av1	C1

Alors qu'avec un modèle binaire, on obtient :



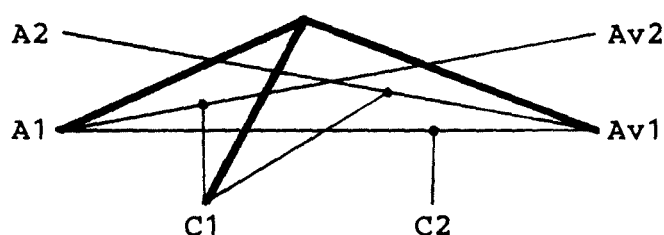
ou encore :

Implique	
Acte	Activité
A1	Av1
A1	Av2
A2	Av1

Est impliqué dans	
Acte	Contrat
A1	C2
A1	C1
A2	C1

Régit	
Contrat	Activité
C2	Av1
C1	Av2
C1	Av1

Ce qui donne, en essayant de recomposer la relation ternaire :



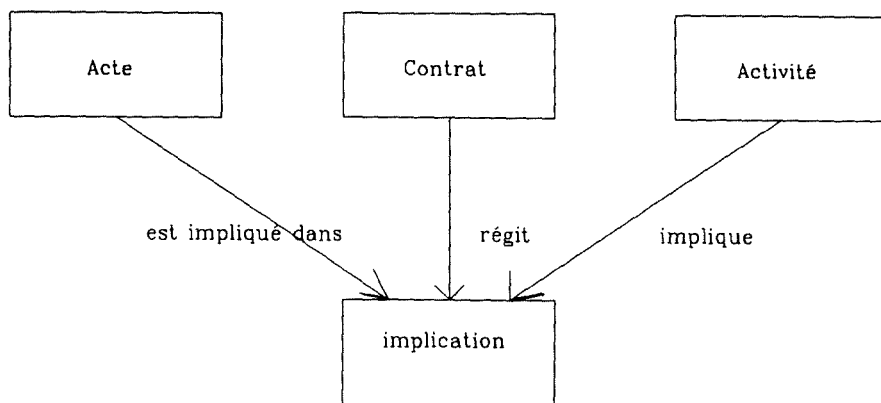
ou encore :

Impliquer		
Acte	Activité	Contrat
A1	Av1	C2
A1	Av1	C1
A1	Av2	C1
A2	Av1	C1

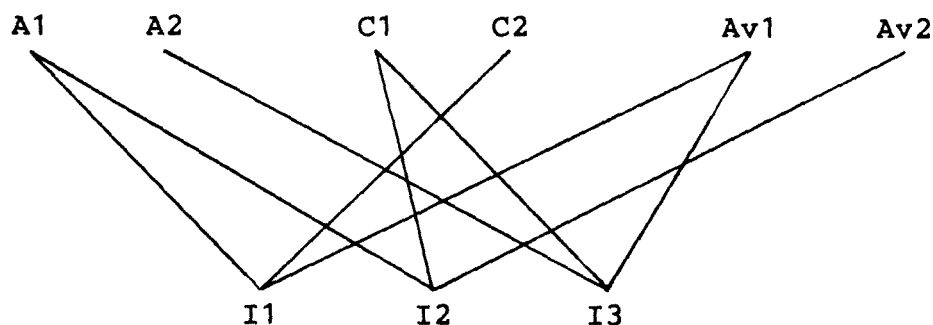
Ce qui constitue un résultat différent de celui obtenu en utilisant un modèle E-R.

Si nous voulons ne pas perdre d'informations et fournir une représentation correcte, il faut dès lors introduire un type d'objet supplémentaire : implication.

On obtient alors :



Le schéma des occurrences nous prouve que cette manière de modéliser est correcte puisque nous obtenons de la sorte :



ou encore :

Est impliqué dans	
Acte	Implication
A1	I1
A1	I2
A2	I3

Régit	
Contrat	Implication
C2	I1
C1	I2
C1	I3

Implique	
Activité	Implication
Av1	I1
Av2	I2
Av1	I3

Cette représentation n'est toutefois pas naturelle. En effet, la relation que nous décrivons est, en soi, une relation ternaire entre l'acte, l'activité et le contrat.

Nous pouvons donc en conclure que les modèles binaires (1 & 2) s'avèrent inadéquats pour représenter ce type de problème.

Les modèles E-R et OPRR, par contre, conviennent mieux dans ce cas, car ils autorisent la représentation de relations multiples.

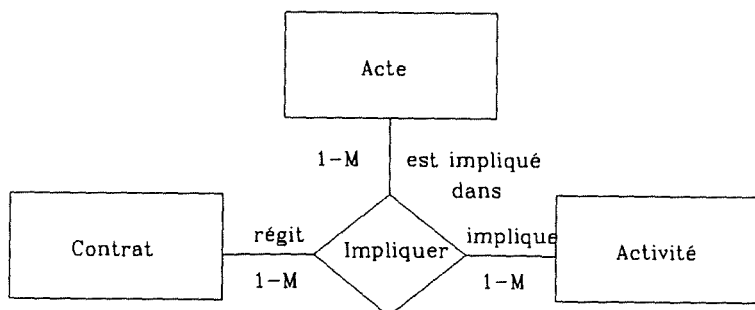
2.2.2.4 Niveau 4 :

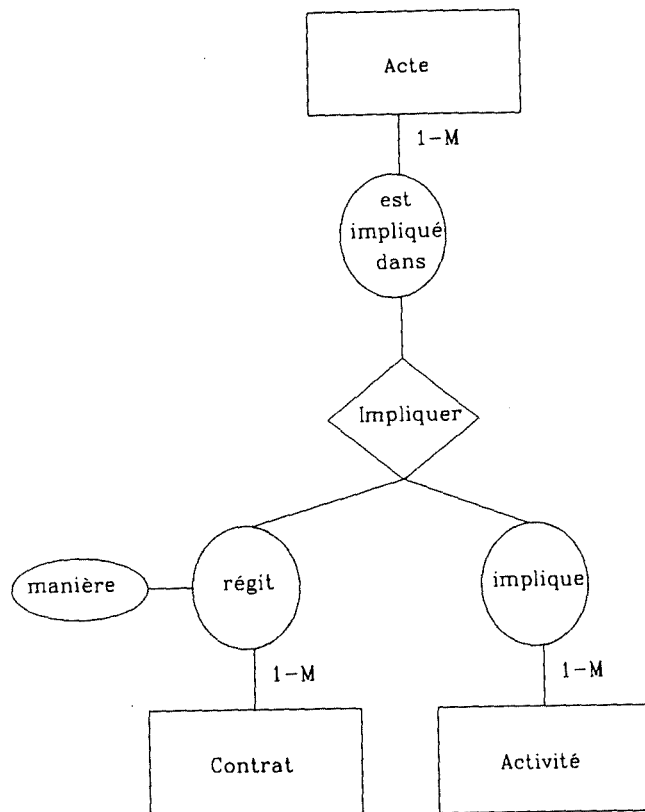
Exemple : Considérons à nouveau l'exemple précédent, mais imaginons cette fois qu'un contrat puisse régir une activité de différentes manières : de manière principale ou accessoire, par exemple, ou de manière directe ou indirecte.

Représentations :

Les modèles de données ERA et OPRR donneraient :

Modèle ERA



Modèle OPRR**Discussion**

Dans ce cas, on peut constater que nous n'avons pu représenter la spécification proposée avec les modèles E-R. Le modèle OPRR s'est avéré nécessaire car il était le seul qui permettait de spécifier une propriété de rôle.

En résumé, les modèles binaires et E-R, bien que suffisants pour certaines spécifications, apparaissent parfois comme inadéquats pour exprimer la richesse de certaines informations.

2.3 LA NOTION DE META-MODELE

Soit un système d'information (S.I.) ; la manière dont une personne voit ce système est appelée schéma-perception ou plus simplement schéma. Un schéma peut donc être décrit comme "un filtre mental pour identifier et extraire un ensemble très réduit de caractéristiques du S.I." [WEL83]. Quand on voit le S.I. grâce à un schéma, on obtient une image.

Différents schémas, appliqués au même S.I., donnent différentes images. L'"externalisation" de ces images sera appelée un modèle "représentationnel" ou plus simplement modèle. Un modèle peut être sous différentes formes "représentationnelles" (texte, liste, matrice, graphique, ...).

Un modèle sous forme OPRR aura comme éléments : des objets, des relations, des rôles et des propriétés.

Un méta-modèle, quant à lui, en utilisant les mêmes composants, n'a pas le même but. Le système à modéliser n'est pas une partie du monde réel, mais bien les éléments d'un autre modèle ou, plus généralement, d'une autre méthode, comme la méthode SAMPO, par exemple.

La différence est que l'information stockée au niveau des instances en utilisant un méta-modèle est un niveau d'abstraction plus élevé que l'information stockée en utilisant un modèle comme le tableau 2.2 l'illustre :

	Contenu décrit par	Exemple de type d'objet	Exemple d'instance d'objet
Modèle	schéma	client	T. Dupond
Méta-modèle	méta-schéma	entité	client

Tableau 2.2

Un méta-modèle peut être très utile pour qui veut comprendre les relations entre les concepts du modèle ou de la méthode modélisés.

CONCLUSION

Nous avons vu dans ce chapitre que les modèles binaires et E-R, bien que suffisants pour certaines spécifications, sont parfois inadéquats pour représenter toute la richesse contenue dans les S.I. actuels.

Dès lors, un modèle plus élaboré, capable de représenter fidèlement toute l'information, est parfois nécessaire. A cette fin, nous avons présenté le modèle OPRR qui possède comme caractéristique principale la possibilité de pouvoir spécifier des propriétés pour les rôles.

Nous avons également étudié la notion de méta-modèle qui nous sera utile par la suite.

CHAPITRE 3

LES OUTILS CASE SUSCEPTIBLES
DE SUPPORTER UNE METHODE

INTRODUCTION

Implanter une méthode comme la méthode SAMPO, c'est-à-dire réaliser un logiciel destiné à en supporter l'usage, n'est pas chose facile.

Pour y parvenir, il faut tout d'abord formaliser la méthode, la modéliser afin de la rendre plus manipulable. Pour ce faire, il existe des outils : les modèles qui ont été présentés dans le chapitre précédent.

Ensuite, il faut informatiser la méthode ainsi modélisée. A cette fin, il existe également des outils, mais informatiques, cette fois. Ils portent le nom d'outils CASE.

Le but du présent chapitre est d'en faire une présentation sommaire et, dans le même temps, de clarifier une terminologie en la matière parfois assez confuse.

3.1 LES MODELES

Tout travail de spécification exige, à l'heure actuelle, un grand effort de réflexion. La personne qui réalise un tel travail doit en effet examiner un système selon différents points de vue. Elle doit manipuler des concepts, en créer des nouveaux, étudier les implications de ses hypothèses et, par un processus itératif, améliorer ou remettre en cause les constructions abstraites qu'elle a pu élaborer.

Pour ce faire, cette personne se sert par conséquent de son intelligence, mais aussi souvent d'un papier, d'un crayon et d'une gomme. Elle a besoin d'écrire ses idées pour s'en souvenir ultérieurement. En outre, il peut arriver que les mots ne lui suffisent pas, ne soient pas assez explicites, soient trop lents ; elle fait alors appel à des dessins ou à des schémas pour représenter les relations et les dépendances structurelles entre les concepts manipulés. Ses idées évoluant rapidement, une gomme lui est indispensable.

Pour pouvoir relire ses notes, ses schémas et ses dessins, le spécificateur s'impose, par ailleurs, de respecter des règles précises dans sa façon de représenter ses idées et de les classer. De plus, lorsque plusieurs personnes travaillant sur un même projet ou sur des projets adjacents veulent se comprendre, elles sont obligées de parler le même langage et doivent dès lors respecter les mêmes formalismes.

Les deux premiers chapitres de ce mémoire ont fait état à plusieurs reprises de formalismes de ce genre, aussi bien graphiques (tels le graphe de discours présenté au point 1.4.3.1 ou les formalismes des modèles binaires, E-R et OPRR du chapitre 2) que non graphiques (telles les tables de l'analyse du discours introduites au point 1.4.3.3).

D'autres formalismes sont également utilisés par les informaticiens, les ingénieurs ou les mathématiciens, à d'autres fins. Il suffit de penser aux organigrammes, aux schémas de bases de données, aux réseaux de Petri, ...

L'utilisation de modèles, c'est-à-dire de formalismes auxquels sont associés des règles d'utilisation, est donc très répandue chez ceux dont le métier est de spécifier, concevoir et mettre en oeuvre des systèmes.

Mais un papier, un crayon et une gomme sont des instruments parfois peu commodes. Il est plus facile d'écrire un texte à l'aide d'un logiciel de traitement de textes qu'avec un stylo. De même, la personne qui élabore des spécifications peut gagner du temps si elle peut disposer d'un "logiciel de traitement de modèles".

Les avantages d'un tel outil sont nombreux [SAU88]. Il fait gagner du temps dans l'élaboration des spécifications, parce qu'il automatise un certain nombre de tâches ancillaires comme, par exemple, le tracé des liens et le tracé des contours des objets, si on travaille avec un formalisme graphique.

Il permet en outre des modifications très aisées de ces spécifications et apporte une aide substantielle en imposant le respect des règles inhérentes au formalisme utilisé. Par sa convivialité et sa facilité d'emploi, il peut encourager celui qui l'utilise à formaliser son travail. Lorsque plusieurs personnes l'utilisent pour un même projet, cet outil peut améliorer sensiblement le rendement et la qualité du travail en résolvant (du moins en partie) les problèmes de communication entre ces personnes.

Enfin, la majorité des spécifications étant destinées à des clients, ces derniers doivent pouvoir comprendre ce que le spécificateur leur propose, dans un langage qui leur soit accessible. Dans pareils cas, l'utilisation de modèles se révèle souvent être un moyen de communication adéquat, pour autant bien sûr que ces modèles soient simples à élaborer, possèdent un grand pouvoir d'expression et soient aisément compréhensibles par les deux parties en présence.

3.2 LES METHODES

C'est en s'appuyant sur des modèles que sont nées les grandes méthodes actuelles de spécification de systèmes d'information et, plus généralement, de développement de systèmes d'information. Comme en témoigne la figure suivante [GIA88], il existe, en effet, des méthodes pour chacune des phases du traditionnel cycle de vie d'un système d'information.

Domaine d'application Phases du cycle de vie d'un S.I.	Fonctionnel	Temps réel	Système distribué	Gestion
1. Analyse des besoins	SADT		Erae	COCOMO
2. Spécification	VDM	State-Charts	Toccata	SAMPO
3. Conception		SDL		Merise
4. Codage	LISP	Chill	C Ada	IDA
5. Validation		MacCabe	Mack	
6. Maintenance	Musa	Littlewood	Halstead Verrall	Cobol Goel-Okumoto

Figure 3.1

Ces méthodes peuvent en fait être définies comme étant "la démarche intellectuelle d'une personne ou d'un groupe de personnes ayant pensé des modèles (des outils) en fonction d'utilisations précises, mais aussi de leur personnalité propre, de leur sensibilité dans l'appréhension des choses" [SAU88].

Il en résulte qu'il n'existe pas, à proprement parler, de bonne ou de mauvaise méthode, chacune ayant ses propres objectifs et tentant de les réaliser au mieux.

Prenons une comparaison dans le domaine des moyens de transport. On dispose, à l'heure actuelle, pour se rendre d'un point à un autre, de nombreuses possibilités, allant du tram bruxellois à la location d'un jet privé. Cependant, chaque moyen de locomotion a ses particularités ou ses impossibilités en fonction du trajet à accomplir.

De la même manière, les méthodes, bien qu'ayant parfois certaines fonctionnalités en commun, ont chacune leur champ d'application privilégié. En ce qui concerne SAMPO, par

exemple, ce champ est constitué par l'étude des communications ou des discours.

Tout comme c'était déjà le cas pour les modèles qui leur servent de base, ces méthodes sont souvent, pour être réellement efficaces, supportées par des outils informatiques.

Dans le cas des modèles, nous avons appelé ces outils des "logiciels de traitement de modèles" (cf. supra). Nous aurions toutefois également pu employer le terme "CASE tools", par analogie au terme "CASE toolkits" qui est employé par la littérature pour faire référence aux "logiciels de traitement de méthodes". Les initiales "CASE" dans ces expressions ont des significations variées suivant les auteurs. La plus fréquente est "Computer Aided Software Engineering", mais le "S" est parfois pris pour "System" ou "Specification" (dans le cas où la méthode est une méthode de spécification, bien sûr) et le "E" pour "Environment".

De manière plus formelle, un CASE toolkit est reconnu par la littérature comme étant "une collection d'outils logiciels intégrés destinée à fournir une assistance automatisée pour une phase du développement d'un logiciel (ou d'un système d'information). Ces outils utilisent une base de spécifications (1) commune pour ranger toutes les informations techniques ou de gestion de projet, partagent une même interface utilisateur et une même interface entre eux" [CLU89].

Un CASE toolkit est par conséquent d'autant meilleur que les outils qui le composent sont de qualité et que leur intégration a été réussie. Il possède les avantages et les inconvénients de ces outils et doit permettre de manipuler de façon cohérente des données hétérogènes dans un même environnement.

3.3 LES METHODOLOGIES

Les données à manipuler sont d'ailleurs encore plus hétérogènes si elles proviennent de l'utilisation d'une méthodologie plutôt que de celle d'une méthode. Stricto sensu, le terme "méthodologie" ne devrait pas faire partie du vocabulaire informatique puisqu'il signifie "étude des méthodes" [LAR87]. La

(1) traduction du terme anglais "repository".

littérature informatique lui octroie cependant une définition toute autre puisqu'une méthodologie est, d'après elle, "une collection de méthodes basées sur une philosophie commune et qui couvre l'entièreté du cycle de vie d'un système d'information" [ORR89].

Il importe donc de ne pas confondre, ce qui est fait de manière relativement fréquente, les termes méthode et méthodologie. Ce dernier terme recouvre en effet une approche beaucoup plus globale que le premier, puisqu'elle ne se limite pas à une phase du cycle de vie d'un système d'information, mais s'étend de l'analyse des besoins à la maintenance du système.

Les données à manipuler lorsqu'on utilise une méthodologie sont par conséquent également beaucoup plus hétérogènes encore que lorsqu'on utilise une méthode. Il suffit, pour s'en rendre compte, de considérer que ces données peuvent être des chiffres, des graphiques, des algorithmes, des modules, des jeux de tests, de la documentation, ...

Dès lors, plus encore que pour les modèles ou les méthodes, le fait de pouvoir disposer d'un support informatique de "traitement de méthodologies" s'avère important et, plus que jamais, son caractère intégré en fera sa valeur.

Un support de ce type est mentionné dans la littérature sous l'intitulé "CASE workbench" et défini comme étant "une collection d'outils logiciels intégrés destinée à fournir une assistance automatisée pour l'ensemble des phases du cycle de vie d'un logiciel (ou d'un système d'information). Ces outils partagent un ensemble d'hypothèses communes sur ce cycle de vie et sur la méthodologie automatisée, utilisent une base de spécifications (1) commune pour stocker toutes les informations techniques ou de gestion de projet, transfèrent automatiquement les informations d'une phase de développement à la suivante, partagent une même interface utilisateur et une même interface entre eux" [CLU89].

Le but de ces CASE workbenches est d'arriver à fournir comme résultat un système opérationnel et correctement documenté.

(1) traduction du terme anglais "repository".

Schématiquement, ces quelques mises au point relatives au vocabulaire utilisé par la littérature en la matière, peuvent se représenter comme suit :

	Support théorique	Support informatique
Entièreté du cycle de vie d'un S.I.	Méthodologie	CASE workbench
Phase du cycle de vie d'un S.I.	Méthode	CASE toolkit
Partie de phase du cycle de vie d'un S.I.	Modèle	(CASE) tool

Figure 3.2

Attardons-nous à présent quelque peu sur la partie qui nous intéresse dans le cadre de ce mémoire, à savoir celle qui a trait aux méthodes et aux outils destinés à les supporter : les CASE toolkits. Nous allons en effet devoir, dans les chapitres qui vont suivre, implanter la méthode SAMPO.

3.4 LES CASE TOOLKITS SPECIFIQUES ET GENERIQUES

Comme nous l'avons déjà dit, il n'existe pas à proprement parler de bonne ou de mauvaise méthode. Bien qu'utilisant en gros les mêmes concepts sur des formalismes plus ou moins semblables et ayant des fonctionnalités voisines, elles sont en effet peu comparables étant donné que chacune d'entre elles s'attache à tel ou tel aspect particulier du réel, possède son domaine d'application privilégié et tente de réaliser au mieux des objectifs qui lui sont propres.

Choisir une méthode se révèle par conséquent être une tâche relativement ardue puisqu'il faut en trouver une qui offre la meilleure adéquation possible entre ses objectifs propres et ceux de son utilisateur potentiel.

D'autre part, utiliser trente-six méthodes, n'aide certainement pas à la communication et à la mise en place d'un

fil conducteur auquel vont pouvoir se raccrocher les divers intervenants d'un projet.

Dès lors, il peut arriver qu'une personne, de par les attentes qu'elle en a, ne trouve pas sur le marché la méthode qu'elle recherche. Deux possibilités s'offrent alors à elle. La première consiste à adapter à ses besoins et de manière plus ou moins profonde une méthode déjà existante ; la seconde, à concevoir et à mettre en oeuvre une nouvelle méthode.

C'est cette voie qu'ont suivie, par exemple, Mr. K. Lyytinen et son équipe, lorsqu'ils ont décidé de donner le jour à la méthode SAMPO. Cette méthode, comme cela a déjà été mentionné, est destinée à remédier à certaines lacunes présentes dans les méthodes de spécification de systèmes d'information en mettant l'accent sur la nature sociale des activités de bureau.

Par ailleurs, il faut toujours garder en mémoire qu'une méthode ne peut jamais rester figée. Elle doit se transformer, se développer, évoluer, s'adapter en fonction des changements qui ont lieu dans les raisons pour lesquelles on l'utilise, la manière dont on l'utilise, les personnes qui l'utilisent, ...

Ces différentes considérations concernant les méthodes ont poussé les sociétés qui opèrent sur le marché des logiciels de traitement de méthodes à proposer des CASE toolkits de deux types.

D'une part, elles offrent des CASE toolkits spécifiques, destinés à servir de support informatique à une méthode bien particulière, fixe et non évolutive. L'implantation de la méthode est, dans ce cas, intégrée à l'outil qui la supporte, et le tout est vendu "clé en main".

D'autre part, elles offrent également, mais généralement depuis moins longtemps, des CASE toolkits génériques. Il s'agit d'environnements informatiques livrés, soit sans méthode, soit avec une méthode à laquelle il est relativement facile d'en substituer une autre, au choix de l'utilisateur. L'implantation de la méthode -s'il y en a une- est, dans ce cas, séparée de l'outil qui la supporte.

Avant que l'utilisateur final ne puisse utiliser un outil de ce type, une étape intermédiaire est donc nécessaire, qui vise à implanter la méthode désirée au moyen du CASE toolkit. Cette étape peut être menée à bien par le concepteur de la mé-

thode lui-même, par l'utilisateur de la méthode ou par un intermédiaire comme, par exemple, une société de services en informatique.

Graphiquement, la démarche à suivre se présente par conséquent comme suit :

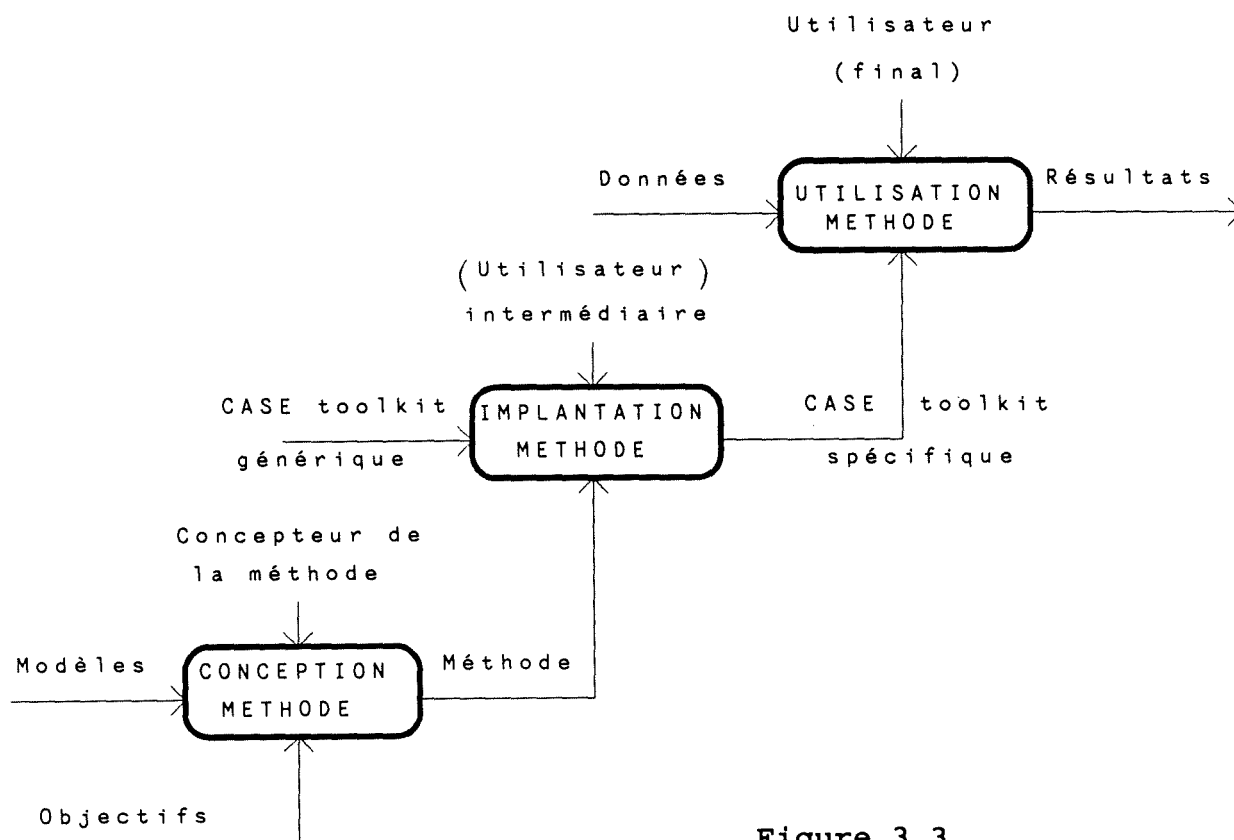


Figure 3.3

3.5 QUELQUES CARACTERISTIQUES DES CASE TOOLKITS GENERIQUES

L'apparition de ce nouveau type d'outils sur le marché peut être considérée sans conteste comme un pas en avant dans le domaine de la spécification et du développement de systèmes assistés par ordinateur : un pas comparable à celui qui a été fait en matière de systèmes experts lorsque sont apparus les premiers logiciels séparant l'expertise et la manière d'utiliser cette expertise.

Ces outils permettent en effet, de par leur caractère générique et instanciable, de doter une méthode d'un support informatique en un temps relativement réduit et à un coût relativement faible. L'implantation complète d'une méthode au moyen de langages de programmation traditionnels nécessiterait

en effet un temps et des efforts de loin plus considérables puisque cela reviendrait, en fait, à réaliser un CASE toolkit spécifique. Ces outils fournissent par conséquent, dès à présent, un réceptacle pour les prochaines générations de méthodes.

Par ailleurs, de par leur caractère configurable et paramétrable, ils peuvent également s'adapter à l'évolution que subissent les méthodes qu'ils supportent, permettant de la sorte à ces dernières de réaliser toujours mieux les objectifs qu'elles se sont fixés.

Enfin, l'extensibilité dont ces outils font preuve est propice à une certaine réutilisabilité lors de l'implantation d'une nouvelle méthode.

Les CASE toolkits génériques se veulent donc des outils stables permettant à tout qui le désire de construire effectivement SON support informatique cohérent, durable, évolutif et performant.

C'est la condition d'ailleurs pour que l'utilisation de cet outil ne soit pas une charge ou une contrainte mais une composante essentielle de la productivité et de la qualité du travail accompli.

Dans le chapitre qui va suivre, nous allons avoir recours à un outil de ce type pour implanter la méthode SAMPO. Si l'on se réfère à la figure 3.3, notre rôle sera, par conséquent, celui de l'intermédiaire.

Bien que nous soyons conscients que les CASE constituent un sujet que nous sommes loin d'avoir épuisé et qui mériterait, à lui seul, qu'on lui consacre un mémoire entier, nous avons décidé d'en rester là dans le cadre de ce chapitre introductif. Nous avons, en effet, présenté tous les éléments qui nous sont nécessaires pour mener à bien notre tâche.

Le lecteur qui désirerait poursuivre son étude dans le domaine, pourra avantageusement consulter [ORR89] pour avoir une idée des principales méthodes utilisées à l'heure actuelle, [CLU89] et [DIE89] pour savoir quels sont les principaux CASE toolkits et CASE workbenches disponibles sur le marché, et [BRE89] et [PUT88] pour connaître l'avis d'utilisateurs de ces outils.

Il trouvera par ailleurs, en annexe A1, deux listes de critères destinées à faciliter son choix parmi les différents produits offerts.

CONCLUSION

Maintenant que nous avons une idée plus précise de ce qu'est une méthode et que nous en avons analysé une dans le premier chapitre, maintenant que nous réalisons mieux ce qu'est un modèle et à quoi il peut servir, et que nous en avons donné des exemples dans le deuxième chapitre, maintenant que nous percevons ce qui se cache derrière les termes CASE, CASE toolkit, CASE workbench, etc, grâce à ce troisième chapitre, nous pouvons nous consacrer à la réalisation du but que nous nous sommes fixés dans l'introduction, à savoir implanter la méthode SAMPO après l'avoir préalablement modélisée.

C'est à la poursuite de cet objectif que seront employés les trois prochains chapitres de ce mémoire.

CHAPITRE 4

MODELISATION ET IMPLANTATION DE LA METHODE SAMPO

INTRODUCTION

Dans le chapitre un, il a été question de la méthode SAMPO. Cette méthode de spécification de systèmes d'information dans un contexte de bureau, a comme caractéristique principale d'être basée sur l'étude des communications et de la nature sociale des activités de bureau, plutôt que sur les structures d'information et les procédures de traitement.

Le chapitre deux, quant à lui, était destiné à décrire différents modèles au moyen desquels il était possible de modéliser la méthode SAMPO ou, autrement dit, d'en établir un méta-modèle.

Le chapitre trois enfin, avait pour but de donner un aperçu des outils CASE, et plus particulièrement des CASE toolkits génériques. Ces outils informatiques d'une nouvelle génération sont en effet capables de simplifier grandement la phase d'implantation d'une méthode.

Maintenant que nous avons à notre disposition tous ces éléments, et comme nous venons de le dire dans la conclusion du chapitre précédent, il nous est désormais possible de nous attaquer à la réalisation proprement dite de notre objectif, c'est-à-dire modéliser la méthode SAMPO et l'implanter via l'implantation de son méta-modèle.

Ces deux sous-objectifs constitueront respectivement les sujets des sections 4.1 et 4.2.

Afin de vérifier la validité de l'implantation réalisée, la section 4.3 sera, quant à elle, consacrée à l'introduction d'un exemple et à l'examen du résultat obtenu suite à cette introduction. L'exemple choisi aura trait à une partie de la modélisation d'une firme de vente par correspondance : la firme Petitpas.

4.1 ELABORATION DU META-MODELE DE SAMPO

Avant de passer à l'élaboration du méta-modèle proprement dit, une décision s'impose ; celle qui concerne le modèle qui va être utilisé pour le représenter.

Nous avons en effet vu au chapitre deux que plusieurs modèles étaient en compétition. Nous avons, malheureusement, également dû constater qu'aucun d'entre eux ne constituait la solution idéale, celle-ci dépendant du système à modéliser.

Dès lors, un choix devait être fait. Ce choix, en ce qui nous concerne, s'est porté sur OPRR et cela principalement parce que Quickspec, le CASE toolkit générique qui sera présenté à la section suivante et qui a été choisi pour supporter le méta-modèle de SAMPO, est basé sur le modèle OPRR.

4.1.1 Réalisation du méta-modèle

Le choix du modèle de représentation étant fait, nous pouvons à présent nous consacrer à la modélisation de la méthode SAMPO. Pour ce faire, la démarche suivie a été la suivante.

Le premier travail que nous avons jugé utile et important d'effectuer a consisté en un recensement des différents concepts manipulés par SAMPO, ainsi que de leurs propriétés respectives. Lors de cette étape, nous avons, par exemple, identifié le concept d'agent et les propriétés qui lui sont associées, à savoir un nom et un type.

Pour chaque concept et chaque propriété rencontrés, nous avons ensuite esquissé une définition. En ce qui concerne le concept "agent", nous l'avons défini comme étant "une personne ou un groupe de personnes qui accomplit un (des) acte(s)".

Bien que souvent largement inspirées par celles que l'on peut trouver dans les publications consacrées à SAMPO, ces définitions n'en sont cependant pas toujours des copies conformes. Ainsi avons-nous tenté de remédier au fait que certains concepts s'étaient vu octroyer, suite à l'évolution de la méthode, deux définitions (légèrement) différentes, alors que d'autres avaient du mal à s'en voir attribuer ne fût-ce qu'une seule complète.

Une fois ces esquisses de définitions terminées, nous avons alors tenté d'identifier les relations qui pouvaient exister entre les concepts ainsi définis. C'est à cette étape qu'est apparue, par exemple, la relation "exécuter" qui relie un agent à un ou plusieurs acte(s).

Les relations obtenues sont caractérisées par le fait que, d'une part, elles dépendent bien sûr des définitions données aux concepts sur lesquels elles reposent mais que, d'autre part, elles influencent également ces mêmes définitions. Une itération a donc été nécessaire jusqu'à l'obtention d'une adéquation satisfaisante entre les définitions des concepts et les libellés des relations. Ainsi, sur base toujours du même exemple, et si l'on part du principe qu'il ne peut y avoir qu'une seule relation "exécuter" au départ du concept d'acte, nous avons été obligés de considérer qu'un agent pouvait également être une machine car il est tout à fait possible qu'un acte (de langage) soit accompli par une machine. La définition du concept "agent" a dès lors dû être modifiée en conséquence.

Arrivés à ce stade, si l'on considère que les concepts recensés ne sont, en fait, rien d'autre que des types d'objets, au sens OPRR du terme, il devient possible de produire ce que nous avons appelé le "Dictionnaire des Types d'Objets". Ce document, proposé en annexe A2, regroupe l'ensemble des définitions des types d'objets ainsi que celles de leurs propriétés respectives.

Par ailleurs, ces mêmes types d'objets, leurs propriétés et les relations identifiées entre eux, peuvent constituer, si on les représente graphiquement selon le formalisme OPRR, une première ébauche du méta-modèle de SAMPO. Première ébauche, car pour être complet, un modèle OPRR doit également spécifier des contraintes de connectivité ainsi que des propriétés pour les types de relations et les rôles qu'il contient.

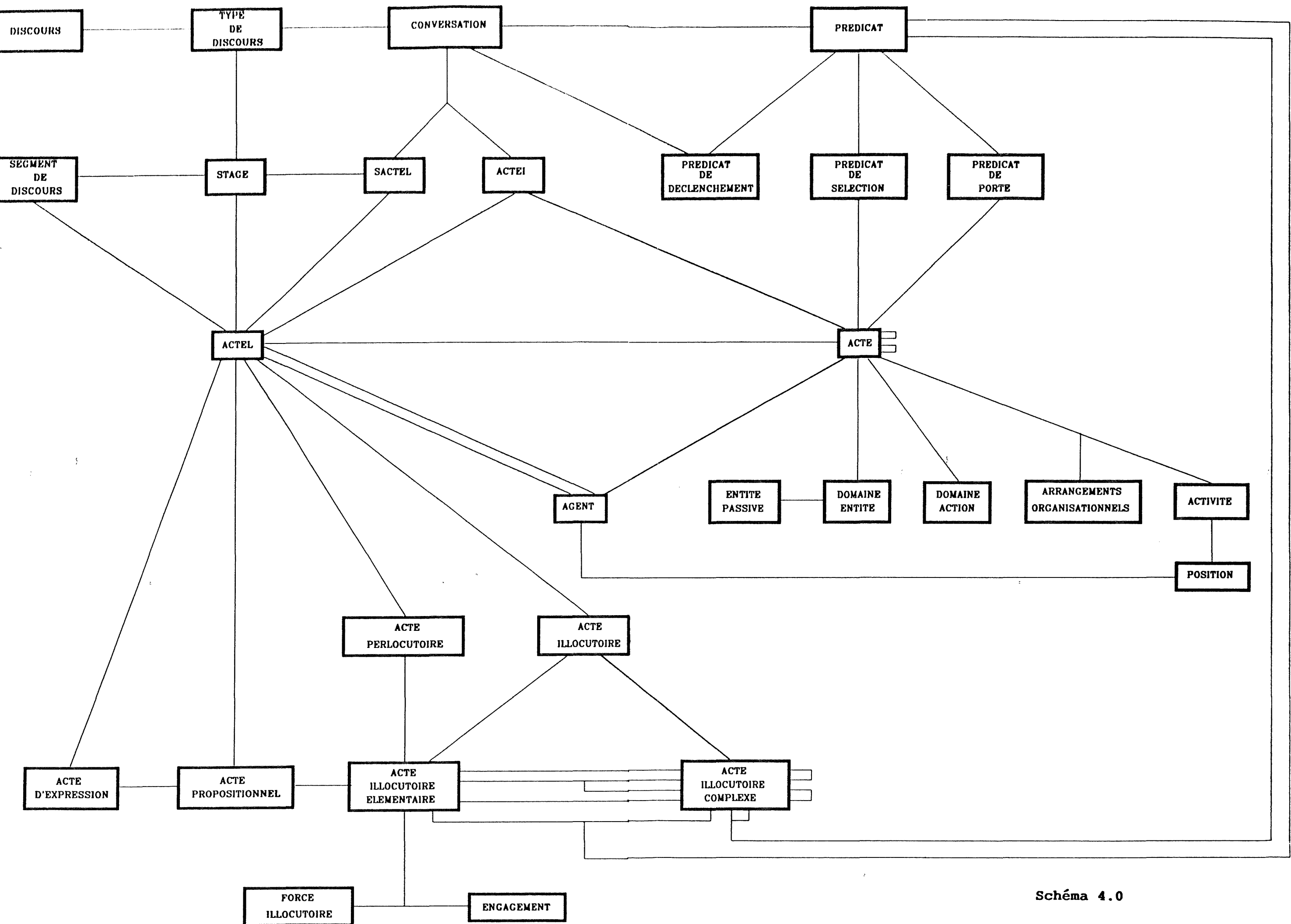
De plus, puisque en matière de modèle, complétude ne rime pas toujours avec exactitude, tout schéma obtenu se doit d'exprimer toute la sémantique attachée au système modélisé. Or il existe des particularités de SAMPO qu'il est impossible de modéliser dans les schémas si on se contente d'utiliser les primitives offertes par OPRR.

C'est pourquoi nous avons décidé d'élargir OPRR en lui ajoutant des possibilités offertes par d'autres modèles, à savoir la possibilité de spécifier des contraintes d'intégrité sur les éléments du modèle [BOD89] ainsi que la possibilité de spécifier des relations de généralisation/spécialisation entre types d'objets [HAI88].

Pour qu'aucune ambiguïté ne provienne de ces deux nouvelles constructions, il faut savoir que [BOD89] a défini une contrainte d'intégrité comme étant "une propriété, non représentée par les concepts de base du modèle, que doivent satisfaire les données appartenant à la mémoire du système d'information". [HAI88] pour sa part estime qu'autoriser des relations de généralisation/spécialisation entre types d'objets revient à "admettre qu'un objet puisse appartenir à plus d'un type d'objet", ce qui "du point de vue des propriétés associées aux objets (...) implique également une structure d'héritage de ces propriétés".

Ces deux ajouts effectués, nous avons pu finalement élaborer la version définitive du méta-modèle "OPRR+" de SAMPO. Cette version est décrite dans les pages qui suivent et commentée juste après celles-ci.

Notons cependant dès à présent que le volume requis par les schémas nous a incités à construire un schéma "plan" présenté sous l'intitulé schéma 4.0.



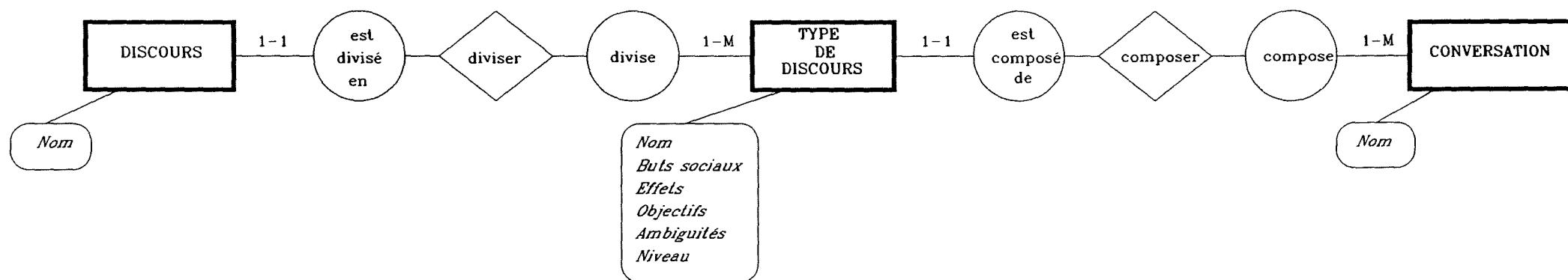


Schéma 4.1

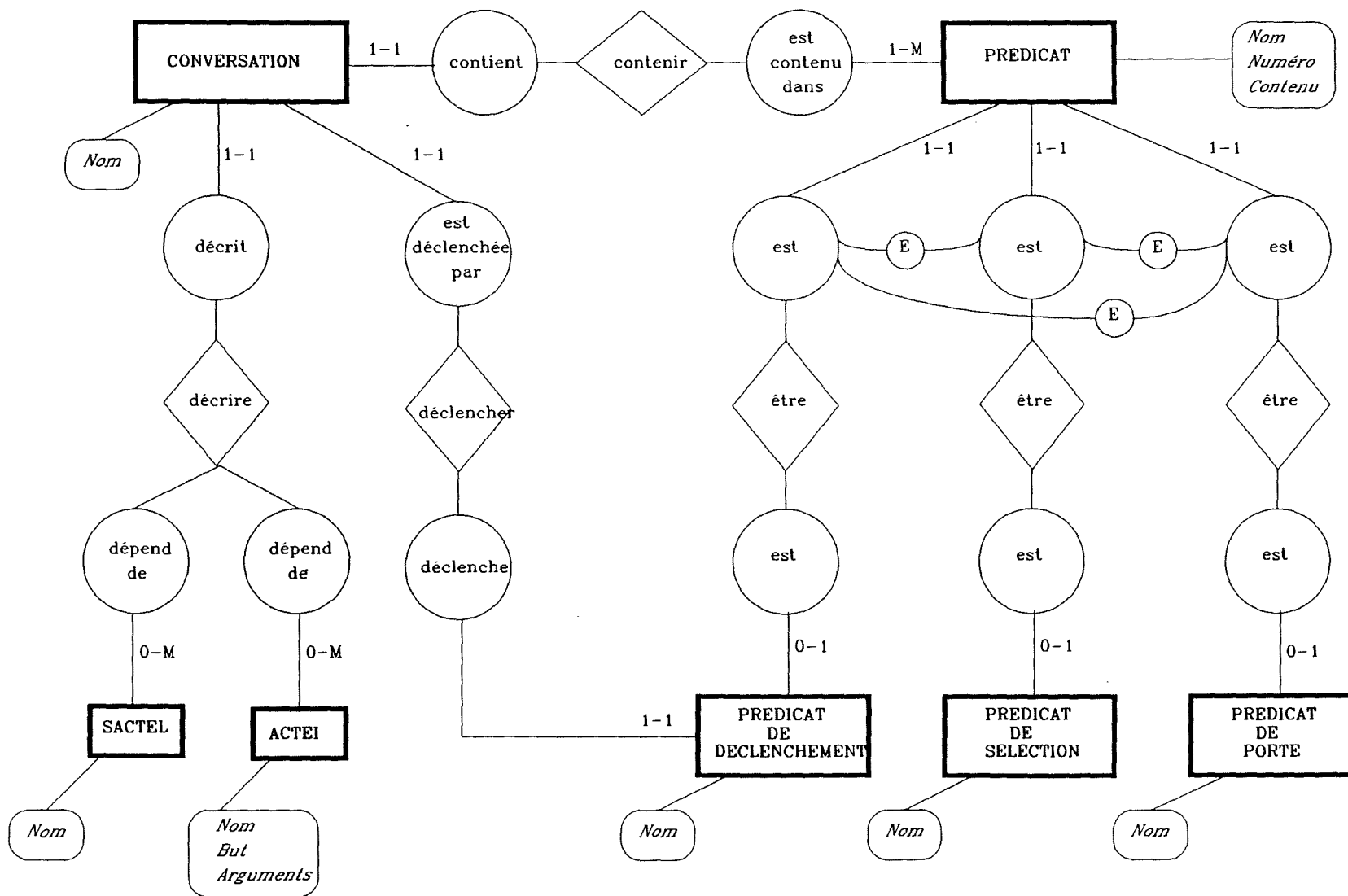


Schéma 4.2

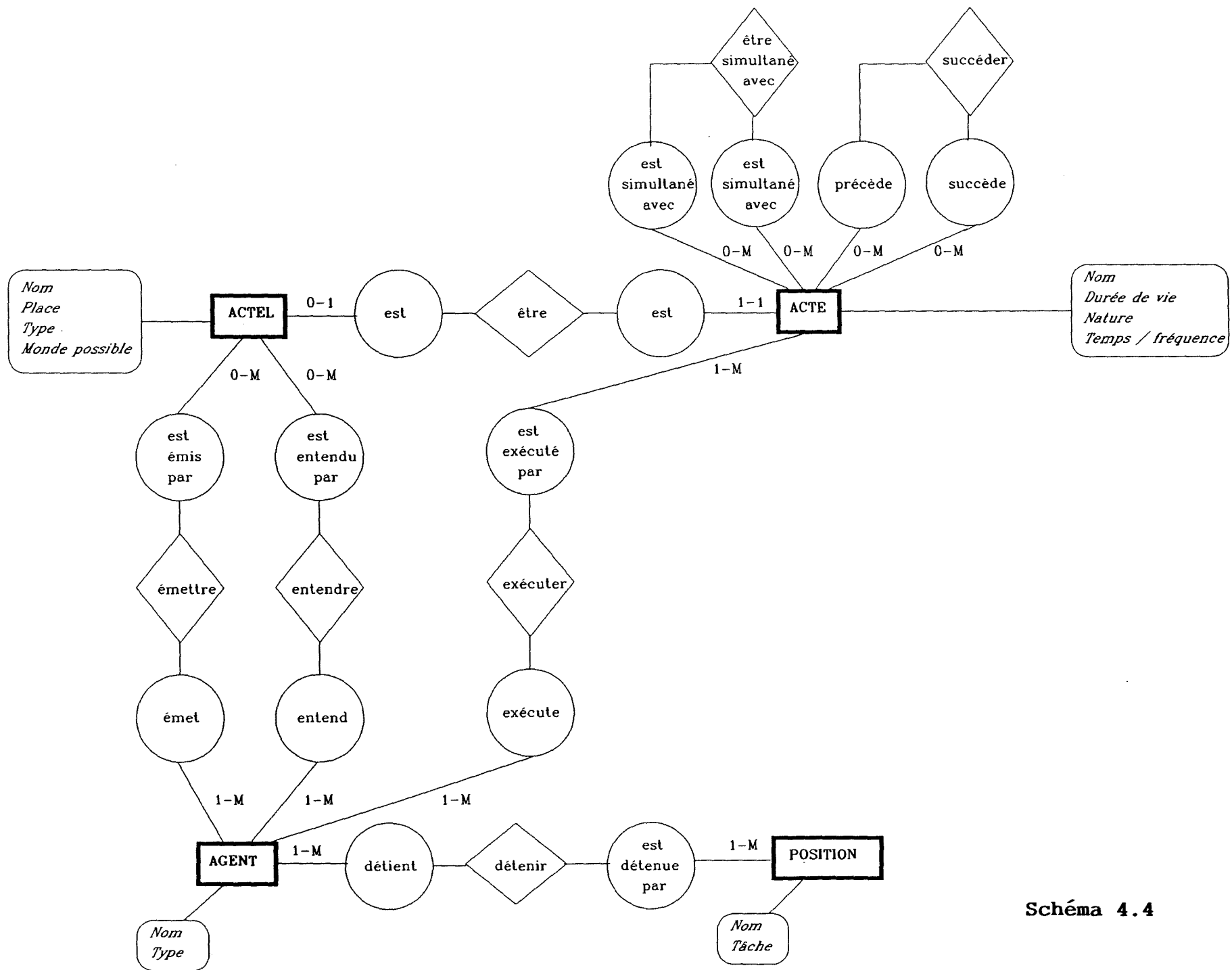


Schéma 4.4

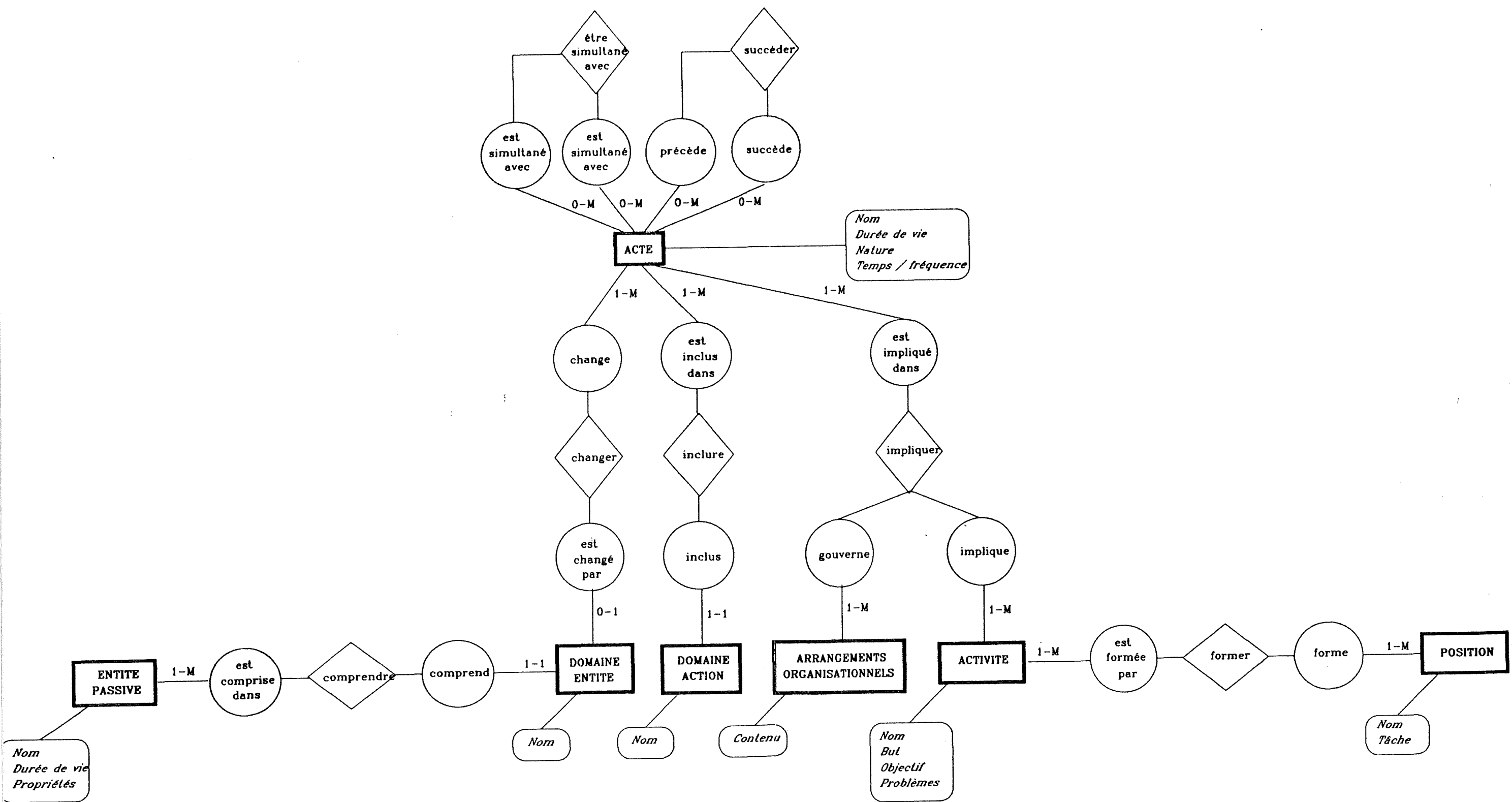


Schéma 4.5

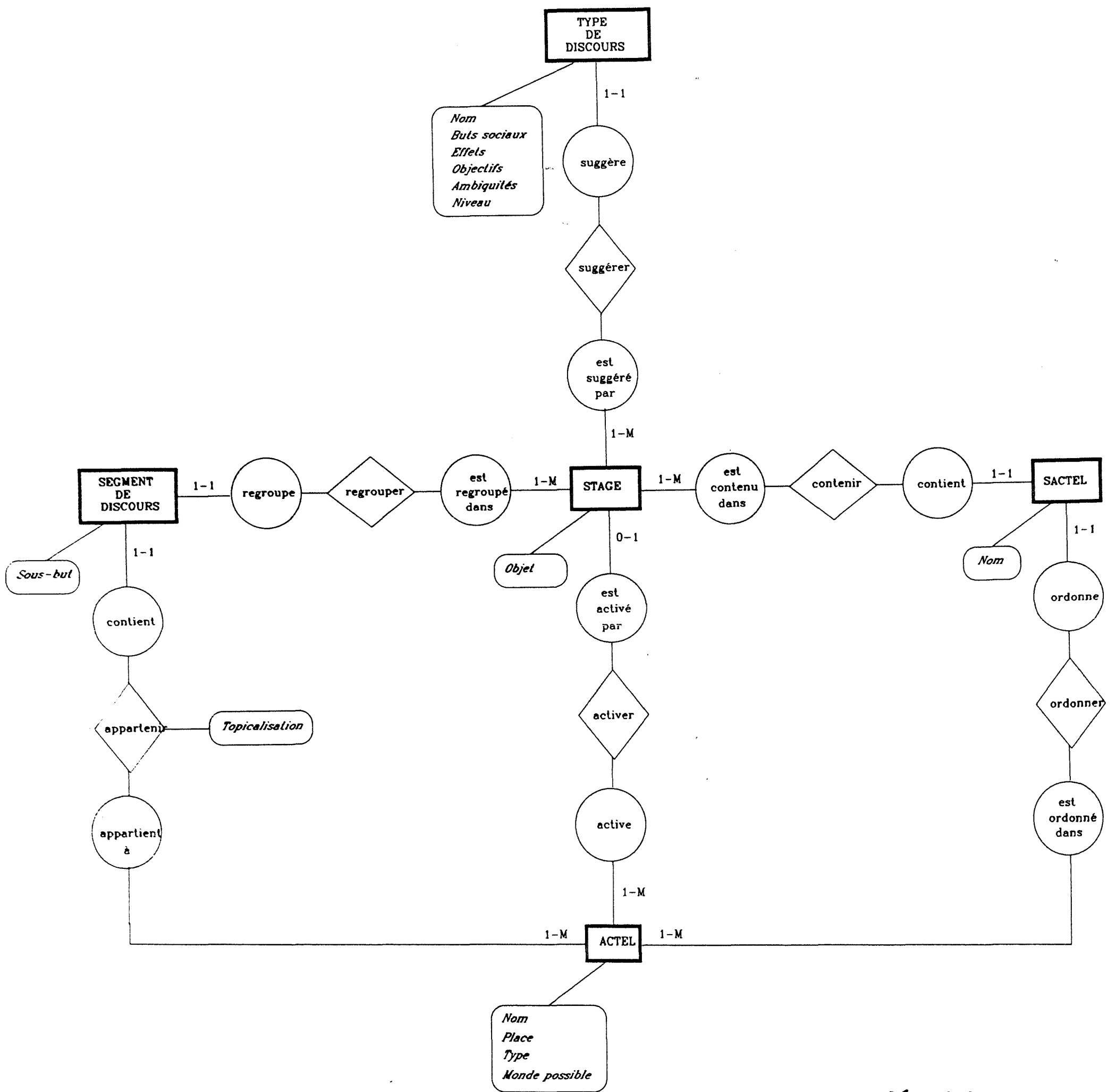


Schéma 4.6

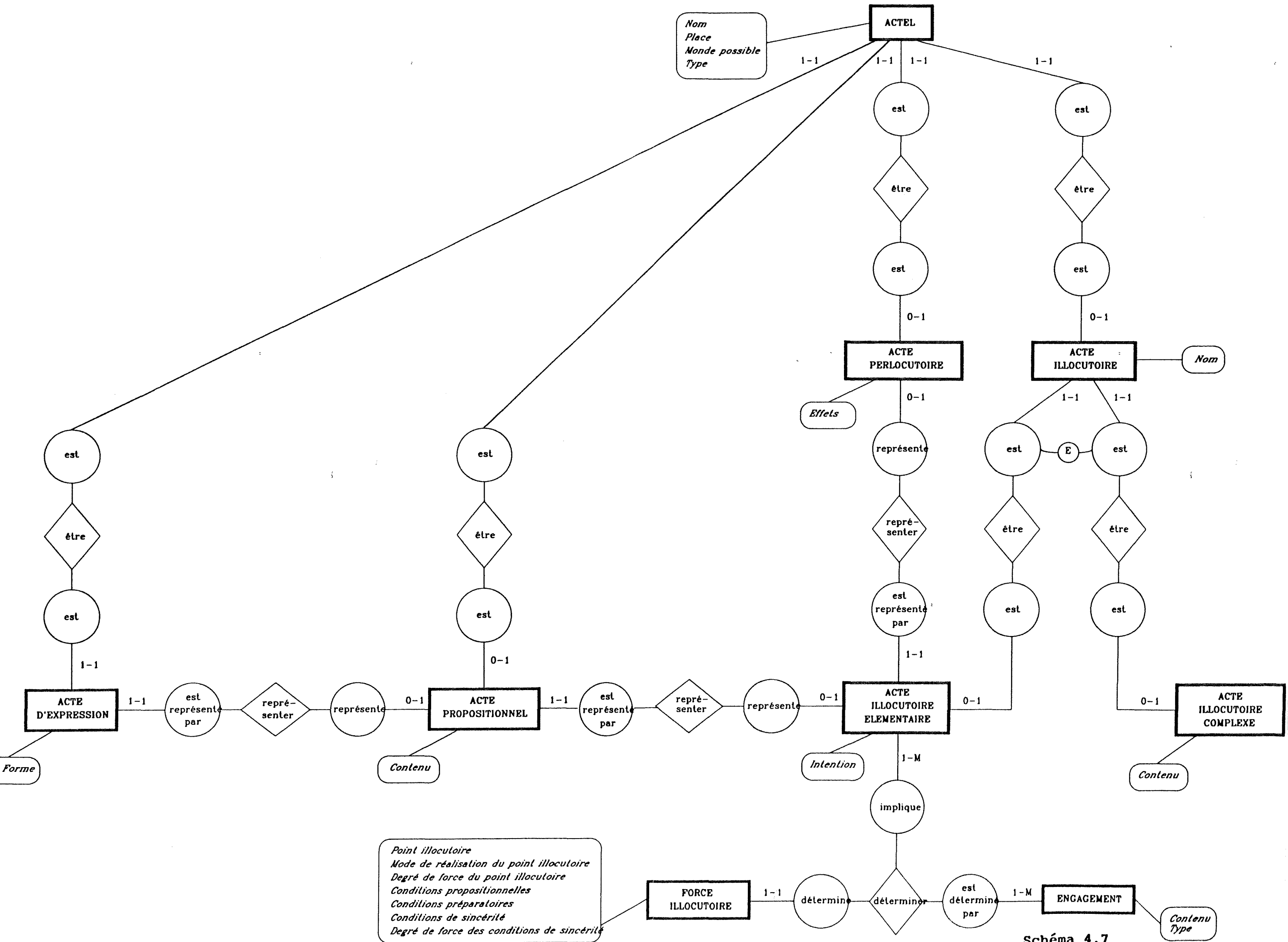




Schéma 4.8

CONTRAINTES D'INTEGRITE.

Tous les noms, dans les schémas, sont identifiants.

1) Exclusions de rôles :

Les exclusions de rôles sont représentées, dans les schémas, par des "E" encerclés.

- Tout prédicat ne peut jouer qu'un seul rôle "est".
- Tout acte illocutoire ne peut jouer qu'un seul rôle "est".
- Tout acte ne peut jouer qu'un seul rôle "est".

2) Attributs :

- La nature d'un acte ne peut être qu'interne ou externe.
- Le type d'un acte de langage ne peut être que virtuel ou non virtuel.
- Le point illocutoire de la force illocutoire ne peut être que assertif, commissif, directif, déclaratif ou expressif.
- Le type d'un engagement ne peut être que faible ou non faible.
- Le niveau d'un type de discours ne peut être que de base et/ou secondaire.

3) Autres :

AIC ::= AIE

AIC ::= not (AIC)

AIC ::= AIC & AIC

AIC ::= P ==> AIC

avec AIC : acte illocutoire complexe

AIE : acte illocutoire élémentaire

P : prédicat

- Si un acte "change" le domaine entité, alors cet acte "est" un acte instrumental.

- Si un acte de langage est de type "virtuel", alors l'(les) engagement(s) qu'il détermine est (sont) de type "faible".
- Si un agent est de type "machine", il ne peut pas exécuter un acte qui "est" un acte instrumental.

4.1.2 Commentaires sur le méta-modèle

Au vu de ces schémas, le premier commentaire que l'on peut faire est que les notions ajoutées à OPRR n'ont en rien été inutiles. Toutes deux ont été nécessaires à la modélisation de SAMPO.

Ainsi, la notion de généralisation a-t-elle été utilisée à quatre reprises pour décrire respectivement un prédicat, un acte, un acte de langage et un acte illocutoire.

La notion de contrainte d'intégrité, pour sa part, nous a permis de spécifier des exclusions de rôles (représentées graphiquement par un E encerclé), des contraintes portant sur les attributs de types d'objets ainsi que d'autres contraintes plus générales encore.

Il s'agissait donc bien là de réelles lacunes sémantiques dans la définition du modèle OPRR.

Bien que portant moins à conséquence, nous avons également constaté dans ce modèle certaines lacunes de type syntaxique. Ainsi avons nous déploré l'absence de conventions graphiques pour représenter les différents types d'attributs rencontrés. Les auteurs d'OPRR auraient pu, par exemple, s'inspirer de [BOD89] et décider de souligner les identifiants, de mettre les attributs facultatifs entre parenthèses et d'ajouter une astérisque à ceux qui sont répétitifs. Une telle convention aurait sans nul doute donné lieu à des schémas plus complets et plus lisibles.

Dans le même ordre d'idées, nous avons également déploré la place prise par les schémas qui résultent de l'utilisation d'OPRR. La représentation graphique des différents éléments du modèle nécessite en effet un espace tel qu'il devient rapidement impossible de modéliser un système un tant soit peu important sur une seule feuille, même grande. Pour ce qui concerne SAMPO, par exemple, nous avons été contraints de désagréger notre schéma et de le répartir sur plusieurs feuilles. Cette désagrégation va de pair, bien sûr, avec la perte d'une vue globale du système.

Dès lors, comme cela a été mentionné précédemment, nous avons tenté de pallier cet inconvénient en construisant un schéma "plan" destiné à fournir au lecteur une sorte d'index. Ce schéma reprend la modélisation de l'ensemble du système,

mais sans noms de types de relations ni de rôles et sans aucune propriété.

Cette dernière observation ne constituerait cependant pas un défaut majeur du modèle si la place requise était utilisée à bon escient, mais, malheureusement, c'est loin d'être le cas ; la plus grande partie du papier étant destinée à représenter les rôles encerclés. Il en résulte qu'un schéma OPRR, bien que plus volumineux, ne contient en réalité aucune information de plus qu'un schéma ERA équivalent si ce n'est la spécification des propriétés de rôles.

Or, malgré l'utilité théorique de ces propriétés, mise en évidence par les exemples traités au chapitre 2, notre (petite) expérience semble prouver qu'elles ne présentent qu'un intérêt minime et accessoire en pratique. Elles sont en effet très rares, très difficiles à déceler, et peuvent, en outre, être confondues avec des propriétés de types de relations. Il est permis dès lors de se demander quel avantage il peut encore y avoir à employer un modèle "élaboré" comme OPRR en lieu et place de modèles Entité-Relation plus traditionnels.

Force est de reconnaître cependant que OPRR a au moins un avantage, partagé il est vrai par les modèles Entité-Relation, celui d'être "user-oriented". Cette caractéristique nous évite, en effet, de devoir expliquer longuement les schémas obtenus et nous permet de nous limiter à deux commentaires.

Le premier voudrait attirer l'attention du lecteur sur le contraste que l'on peut observer entre d'une part le nombre élevé de types d'objets relevés dans la méthode SAMPO (28) et d'autre part le faible nombre de propriétés que possède chacun de ces types d'objets.

Une solution à ce déséquilibre aurait consisté à considérer certains concepts non pas comme des types d'objets, mais bien comme des propriétés, ou encore comme des types de relations. Dans la plupart des cas, cette solution s'est cependant avérée impossible à mettre en pratique car elle aurait contribué à dénaturer la méthode modélisée. C'est ainsi, par exemple, que le concept "position" aurait pu être considéré comme une propriété du type d'objet "agent" plutôt que comme un type d'objet autonome. Cette alternative ne permettant toutefois pas de conserver le type de relation "former" existant

entre les types d'objets "position" et "activité" a donc dû être abandonnée.

Le second commentaire, plus ponctuel, est quant à lui destiné à justifier l'absence de la notion de MOUVEMENT dans les schémas. Ce concept a, rappelons-le, été défini par SAMPO comme étant "un acte de langage qui active un stage dans un discours". Il constitue donc un cas particulier d'acte de langage et peut, à ce titre, ne pas être considéré comme un type d'objet autonome. C'est la raison pour laquelle nous avons décidé, in fine, de ne pas le mentionner dans les schémas.

Ces commentaires effectués, nous pouvons à présent passer à la phase d'implantation de la méthode SAMPO.

4.2 IMPLANTATION DE LA METHODE SAMPO

Avoir à sa disposition le méta-modèle d'une méthode est utile à tout analyste qui désire employer cette méthode, personne n'en disconviendra. Cela lui permet, en effet, de mieux en appréhender les éléments constitutifs ainsi que leurs inter-relations.

Mais ce qui pourrait être encore plus intéressant pour un analyste serait de pouvoir disposer d'une implantation de cette méthode, c'est-à-dire d'un environnement informatique basé sur le méta-modèle de la méthode et capable de l'aider activement dans sa tâche de spécification.

C'est à la réalisation d'un environnement de ce genre, destiné à supporter la méthode SAMPO que nous allons consacrer les pages qui suivent.

4.2.1 Introduction à un CASE toolkit générique : Quickspec

Etant donné, comme l'a montré le chapitre précédent, le fait qu'un CASE toolkit est généralement destiné à supporter une méthode, la réaction la plus logique lorsqu'on est confronté à un problème d'implantation de méthode consiste à se tourner vers ces outils, ou du moins vers certains d'entre eux.

Nous avons en effet vu que parmi ces outils, certains, les CASE toolkits génériques, étaient conçus de telle façon qu'ils offraient la possibilité de remplacer, de manière relativement aisée, la méthode qu'ils étaient initialement censés supporter par une autre, quelle qu'elle soit. Cette possibilité s'avère évidemment particulièrement intéressante à tous ceux qui désirent implanter une méthode car elle permet de réduire le temps et le travail liés à cette tâche. C'est pourquoi, pour implanter SAMPO, nous avons eu recours à un outil de ce type.

Quickspec, c'est le nom que lui a attribué Meta Systems Ltd, la firme qui le commercialise, est en fait un logiciel fenêtré et destiné à être utilisé sur PC. Il est basé sur une sorte de base de spécifications intégrée et centralisée qui est caractérisée par le fait qu'elle est prévue pour contenir deux niveaux de spécification et dispose à cette fin de deux bases de données distinctes [WEL88b].

Le premier niveau de spécification, qui est également le plus visible, est celui qui est réservé à l'utilisateur final du logiciel, ou, autrement dit, à un analyste. Il y introduira les spécifications du système qu'il veut modéliser et celles-ci seront stockées dans une base de données appelée "base de données cible". Cette base peut contenir, par exemple, des informations du type "Mr. Lyytinen est professeur".

Le second niveau, pour sa part, a été conçu pour recueillir les règles qui vont régir l'introduction des spécifications dans cette base de données cible. On peut y spécifier les types d'objets ou de relations qu'on considère comme légaux, les propriétés ou les rôles autorisés, les contraintes de connectivité, ... Bref, ce niveau contient la description d'un méta-modèle de méthode tel que spécifié au chapitre deux, et la base de données associée porte dès lors le nom de "méta base de données". On y trouvera des informations telles que "Un agent détient une (des) position(s)", pour reprendre l'exemple précédent.

Par défaut, cette base contient toutefois la description du langage PSL (et non celle de SAMPO), permettant de la sorte à un utilisateur final d'introduire des spécifications rédigées dans ce langage.

Au vu de ces caractéristiques, on peut donc assimiler Quickspec, à un CASE toolkit générique dont l'acronyme serait "Computer Aided (PSL) Specification Engineering". Il représente cependant bien plus que cela étant donné la possibilité qu'il offre de supporter d'autres méta-modèles ou langages que PSL. Il suffit, pour ce faire, de remplacer la description contenue dans sa méta base de données par une autre.

Tout méta-modèle ou langage alternatif à PSL ne doit, par ailleurs, satisfaire qu'une seule condition pour être acceptable : avoir été réalisé en utilisant le modèle généralisé de données associé aux produits de Meta Systems Ltd, qui n'est autre que le modèle OPRR (cf. chapitre deux). Une fois la description du méta-modèle réalisée et stockée dans la méta base de données de Quickspec, ce dernier sera en mesure de la supporter et de l'utiliser à un niveau d'abstraction qui est celui des concepts du modèle OPRR. Il sera donc possible de manipuler directement des objets, des relations, des propriétés, ... c'est ce qui vaut d'ailleurs à ce logiciel d'être considéré comme un outil basé sur OPRR.

Ces précisions sur Quickspec ayant été fournies, nous pouvons à présent envisager de substituer au contenu initial de sa méta base de données, notre méta-modèle OPRR de SAMPO. Une telle substitution permettrait à un utilisateur final d'introduire non plus des spécifications PSL, mais bien des spécifications SAMPO (cf. exemple précédent : "Mr. Lyytinen est professeur") dans la base de données cible qui lui est réservée. SAMPO serait de la sorte implanté et Quickspec deviendrait un CASE toolkit générique dont l'acronyme serait désormais "Computer Aided (SAMPO) Specification Engineering".

Un problème subsiste néanmoins. Il a trait au fait que, bien que basé sur OPRR, Quickspec n'est pas encore capable d'enregistrer dans sa méta base de données un méta-modèle décrit sous forme graphique. Des opérations préalables de conversion des graphiques en texte et de compilation de ces textes sont donc nécessaires à l'obtention d'une base de données compatible avec les attentes du logiciel à ce niveau. Ces opérations pourront être réalisées au moyen du Model Definition Manager (MDM) associé à Quickspec.

Avant de poursuivre la présentation de Quickspec en décrivant ses principales caractéristiques susceptibles

d'intéresser les utilisateurs, il apparaît donc nécessaire de présenter tout d'abord brièvement cet outil intermédiaire que constitue MDM.

4.2.2 Présentation de MDM

MDM est un logiciel destiné à transformer un modèle sous forme graphique en son correspondant textuel. Il est basé sur le modèle de données OPRR, c'est-à-dire qu'il accepte des spécifications sous forme OPRR, et génère en sortie une méta base de données conforme aux besoins de Quickspec.

L'obtention de cette base de données requiert néanmoins le passage par différentes étapes [ISD85] où il faut respectivement :

- identifier les éléments constitutifs du système à modéliser et les formaliser (sous forme graphique),
- exprimer cette formalisation en langage MDM,
- exécuter la déclaration MDM pour créer une méta base de données,
- préparer la spécification de la table XT associée,
- inclure cette table dans la méta base de données créée.

Il est à noter, par ailleurs, que l'utilisation de MDM étant liée à une activité de conception, le processus séquentiel décrit ci-dessus peut se transformer en un processus itératif. Ainsi, il arrive que des retours en arrière aux étapes précédentes soient nécessaires afin de modifier des résultats qui y ont été obtenus.

Passons, à présent, à l'explication du contenu de chacune de ces étapes.

Etape 1 : identifier les éléments constitutifs du système à modéliser et les formaliser.

Cette première étape a pour but l'étude des caractéristiques du système à modéliser ainsi que sa formalisation. Elle peut être menée à bien en utilisant un formalisme de type E-R ou OPRR. Il est cependant souhaitable, si possible, de faire vérifier les graphiques obtenus par une personne compétente.

La formalisation des éléments d'un système se révèle, en effet, souvent être une étape cruciale dans le développement de ce système puisqu'intervenant très en amont dans le cycle de vie et pouvant, par conséquent, avoir de nombreuses répercussions sur la suite des opérations.

La formalisation sous forme OPRR d'une partie de la méthode SAMPO pourrait, par exemple, avoir conduit au schéma suivant :

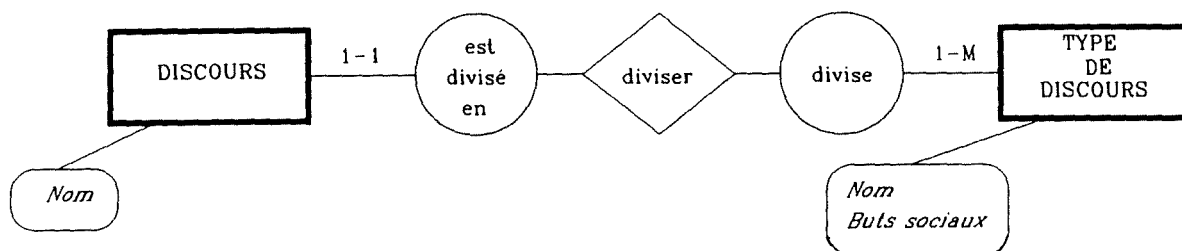


Schéma 4.9

Etape 2 : exprimer cette formalisation en langage MDM.

Une fois ces graphiques vérifiés, l'étape deux va les transformer en instructions MDM et les stocker dans un fichier au moyen d'un éditeur.

L'utilité de l'éditeur à ce niveau est essentielle car il permet non seulement de corriger des erreurs, mais aussi de mettre à jour la description du système. Il s'agit là d'une possibilité réellement intéressante quand on sait que des instructions peuvent être ajoutées partout dans le texte et pas seulement à la fin de celui-ci.

Cette transformation de graphiques en instructions MDM est toutefois régie par certaines règles de syntaxe [MET89a]. Ainsi,

- tout type d'objet doit être défini suivant le format :

```

OBJECT nom_objet ;
CODE valeur_code ;
  
```

où valeur_code est la valeur numérique unique prise par le code.

Les mots clés apparaissent en majuscules et en caractères gras. L'indentation, quant à elle, n'a d'autre prétention que de faciliter la lecture.

- la définition de tout type de relation a le format :

```
RELATIONSHIP nom_relation ;  
  PARTS (nom_rôle spec_conn valeur_code) ... ;  
  COMBINATION nom_rôle liste_objets  
    (WITH nom_rôle liste_objets) ... ;  
  CODE valeur_code ;
```

où nom_rôle est le nom d'un rôle associé à la relation,

spec_conn peut être ONE-ONE, ONE-MANY,
ZERO-ONE, ZERO-MANY,

liste_objets est une liste de types d'objets valides
séparés par des virgules.

- la définition de toute propriété a le format :

```
PROPERTY nom_propriété ;  
  APPLIES-OBJECT liste_objets ;  
  NUMBER-VALUES spec_conn ;  
  VALUES (TEXT/INTEGER/REAL/STRING) ;  
  (IDENTIFIER) ;  
  (CASE-SENSITIVE) ;  
  CODE valeur_code ;
```

De plus,

- toutes les valeurs de code doivent être uniques,
- la propriété "basic-name" doit être présente, avoir une valeur de code égale à 2014 et être de la forme :

```
PROPERTY basic-name ;  
  APPLIES-OBJECT liste_objets ;  
  NUMBER-VALUES ONE-ONE ;  
  VALUES STRING ;  
  IDENTIFIER ;
```

CASE-SENSITIVE ;
CODE 2014 ;

- les relations ne peuvent pas être plus que quaternaires ; ce qui peut constituer une limite pour celui qui emploie des méthodes (automatisées) qui associent un nombre de plus en plus important de types d'objets en vue de formuler, de spécifier ou d'évaluer des spécifications de plus en plus complexes.

Transformé en instructions MDM en respectant ces règles, le schéma 4.9 deviendrait par conséquent :

```

OBJECT Discours;
  CODE 1;

OBJECT Type_discours;
  CODE 2;

RELATIONSHIP Diviser;
  PARTS est_divisé_en      ONE-ONE 3001,
        divide            ONE-MANY 3002;
  COMBINATION est_divisé_en Discours
        WITH divide      Type_discours;
CODE 1001;

PROPERTY Buts_sociaux;
  APPLIES-OBJECT Type_discours;
  NUMBER-VALUES Zero-Many;
  VALUES Text;
  CODE 2001;
  CASE-SENSITIVE;

PROPERTY Basic-name;
  APPLIES-OBJECT Discours,
                  Type_discours;
  NUMBER-VALUES One-One;
  VALUES String;
  CODE 2014;
  IDENTIFIER;
  CASE-SENSITIVE;

```

Etape 3 : exécuter la déclaration MDM pour créer une
 méta base de données.

Après avoir obtenu un fichier source complet, il n'est rien de plus normal que de le compiler. L'étape trois y sera donc consacrée et utilisera pour ce faire un programme appelé MGEN. Le rôle de ce programme est d'analyser les instructions

MDM du fichier source, de détecter les erreurs de syntaxe qu'elles contiennent et de produire un résultat ad hoc.

Dans le cas où aucune erreur n'a été détectée, ce dernier consistera en la création d'une base de données contenant la version objet du fichier source augmentée de définitions automatiquement générées par le système.

Par contre, dans le cas où des erreurs subsisteraient, le résultat consistera en un listing du fichier source parsemé d'autant de diagnostics que d'erreurs trouvées.

Ce listing devra alors être étudié et les corrections appropriées faites dans le fichier source jusqu'à l'obtention de la base de données désirée.

Etape 4 : préparer la spécification de la table XT associée.

Mais, avoir des types d'objets, des types de relations et des propriétés ne sert encore à rien si ces composants ne sont pas visibles à l'utilisateur final du logiciel. L'étape quatre sert par conséquent à définir pour chacun de ces composants une représentation externe ("tag"), c'est-à-dire visible à l'écran.

L'ensemble de ces définitions porte le nom de table XT et est réalisé à l'aide d'un langage auquel, bien entendu, sont associées certaines règles de syntaxe et de sémantique [MET88]. Les principales règles peuvent se résumer comme suit,

- les noms de tags doivent être des strings, de maximum 30 caractères, et délimités par des apostrophes ou des guillemets. Ils sont assignés à tout type d'objet, type de relation ou propriété destiné à être visible à l'utilisateur final.
- pour tout type d'objet, le format est :

```
OBJECT nom_objet ;  
TAG nom_tag ;
```

- pour tout type de relation, le format est :


```

RELATIONSHIP nom_relation ;
    TAG FOR nom_rôle nom_tag WITH
        ROLE position nom_rôle nom_tag,
        ...
    ...

```

où position est un entier positif dont le but est de déterminer l'ordre de présentation.

- pour toute propriété, le format est :

```

PROPERTY nom_propriété ;
    TAG nom_tag ;

```

- tous les tags sont spécifiques à une table XT ; il n'y a aucun besoin d'unicité des noms entre différentes tables,
- dans une table, il est nécessaire que, d'une part, tous les tags de types d'objets soient uniques et que, d'autre part, tous les tags de types de relations et de propriétés soient uniques aussi.

En ce qui concerne l'exemple qui retient notre attention, la table XT devrait donc avoir l'aspect suivant :

```

OBJECT Discours;
    TAG "Discours";

OBJECT Type_discours;
    TAG "Type de discours";

RELATIONSHIP diviser;
    TAG FOR est_divisé_en "Divisé en : " WITH
        ROLE 1 divise "Divise";
    TAG FOR divise "Divise : " WITH
        ROLE 1 est_divisé_en "Divisé en";

PROPERTY Buts_sociaux;
    TAG "Buts sociaux";

PROPERTY Basic-name;
    TAG "Nom";

```

Etape 5 : inclure cette table dans la méta base de données créée.

Arrivés à ce stade, il ne nous reste plus qu'à stocker dans un fichier la table XT réalisée, à compiler ce fichier et à ajouter la version objet obtenue à la base de données créée à l'étape trois. Toutes ces opérations seront effectuées dans la cinquième étape en exécutant le programme XTGEN.

4.2.3 Présentation de Quickspec

Lorsque nous avons vu que Quickspec était capable de supporter d'autres méta bases de données que celle contenant PSL, nous avons su que nous allions pouvoir implanter SAMPO en utilisant ce logiciel.

Maintenant que nous connaissons aussi la procédure à suivre pour construire une nouvelle méta base de données et la substituer à l'ancienne, nous savons également comment nous allons pouvoir réaliser notre implantation.

Avant de se lancer dans ce travail, nous avons toutefois jugé souhaitable de passer en revue quelques caractéristiques de Quickspec susceptibles d'intéresser, non plus l'intermédiaire qui veut implanter une méthode (cf. figure 3.3), mais bien un utilisateur final du logiciel.

Pour rappel, la notion d'utilisateur final fait référence ici à un analyste, ou à une personne désireuse de modéliser des parties du monde réel en utilisant une méthode implantée. Dans notre cas, il s'agit donc d'une personne qui désire spécifier un système d'information de bureau en utilisant notre implantation de la méthode SAMPO.

Il serait intéressant de voir comment cette personne pourra être aidée dans sa tâche par Quickspec et comment elle pourra introduire ses spécifications dans la base de données cible qui lui est réservée.

Signalons cependant que notre but n'est en rien de fournir un mode d'emploi complet de Quickspec. D'autres documents, tels [MET89b], [MET89c], y sont consacrés.

4.2.3.1 Fonctions offertes par Quickspec

Vu sous l'angle d'un utilisateur final, Quickspec constitue un outil informatique destiné à construire, visualiser et mettre à jour les spécifications d'un système dans

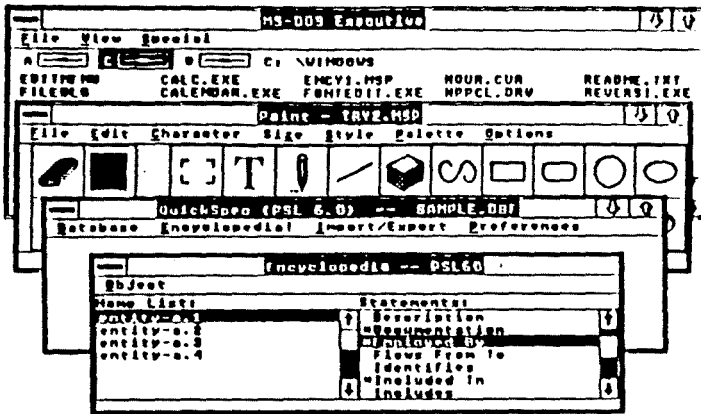


Figure 4.1

Figure 4.2

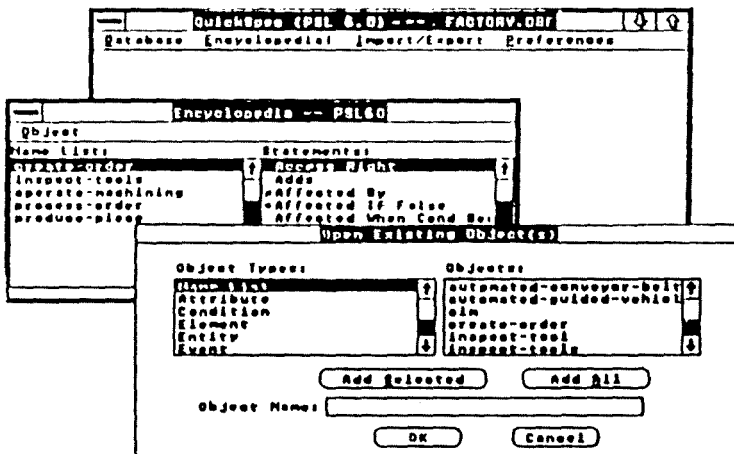
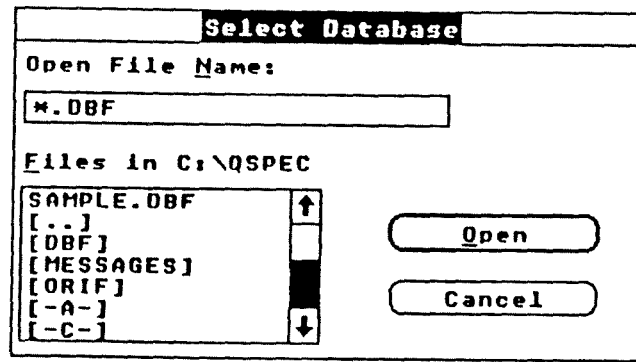
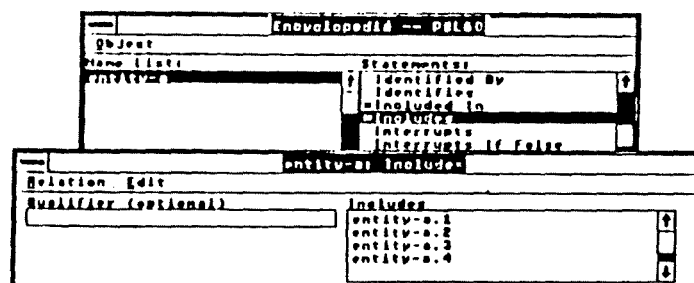


Figure 4.3

Figure 4.4



l'environnement Windows de Microsoft. Il permet, par ailleurs, d'effectuer ces opérations de manière agréable et efficace suite à l'utilisation d'un procédé de sélection d'éléments à l'écran, au moyen d'une souris.

Avec Quickspec, il est donc possible de mémoriser des spécifications (PSL ou autres, cela dépend du contenu de la méta base de données) sans écrire une seule instruction. De plus, une fois ces informations stockées dans la base de données cible, l'utilisateur peut facilement les éditer, en ajouter ou en supprimer.

Enfin, des possibilités d'import et d'export de spécifications en provenance ou à destination d'autres outils commercialisés par Meta Systems Ltd sont également disponibles.

4.2.3.2 Atouts de Quickspec

Pour réaliser au mieux ces fonctions, c'est-à-dire de manière fiable et agréable pour l'utilisateur, Quickspec possède un certain nombre d'atouts :

- il est destiné au monde des PC's, ce qui lui assure un maximum de portabilité et d'efficacité.
Plusieurs analystes peuvent, par exemple, travailler en même temps à différentes parties d'une même spécification. Il suffit pour ce faire que chacun d'entre eux dispose d'un PC. Après quoi, il est possible d'intégrer les résultats individuels, avec la fonction d'export, pour obtenir la solution globale désirée.
- son interface utilisateur, Windows de Microsoft, est reconnue de manière unanime comme étant très conviviale (cf. figure 4.1).
Cela est dû à l'emploi de symboles visuels tels que les fenêtres, les icônes, les menus déroulants, les barres de défilements, les boîtes de dialogues, ... L'usage de la souris contribue aussi, souvent, à faciliter et à accélérer les opérations à effectuer.
- il fournit une base de données pour stocker et mettre à jour les spécifications (cf. figure 4.2).

Cette base assure l'intégrité des spécifications stockées en les protégeant contre les duplications, violations de règles de langage, inconsistances, ... Elle permet en outre de mémoriser l'information sous un format indépendant de toute représentation.

- il donne un accès direct aux informations contenues dans la base de données et autorise l'utilisateur à modifier facilement et rapidement ses entrées (cf. figure 4.3).

Un seul "click" sur la souris suffit pour faire afficher à l'écran ou modifier des objets contenus dans la base, des propriétés, des relations ou des rôles. Il n'est plus nécessaire de maîtriser des procédures d'édition complexes, de taper des commandes au clavier ou de se souvenir de paramètres compliqués.

- il vérifie les informations entrantes de façon à produire des spécifications complètes et cohérentes (cf. figure 4.4).

Les types d'objets et les types de relations résultant d'une sélection à l'écran, il devient impossible d'introduire des relations incomplètes ou illégales. Quickspec se limite dès lors à vérifier que les noms introduits sont syntaxiquement et sémantiquement corrects avant de les mémoriser dans la base de données.

- il réduit le temps d'apprentissage de la méthode contenue dans la méta base de données.

Une fois de plus, le système de sélection à l'écran évite de s'encombrer l'esprit avec des mots clés, des caractères de ponctuation, ...

4.2.3.3 Les règles liées aux entrées

Malgré que le temps d'apprentissage soit réduit, il n'en reste pas moins vrai que certaines règles subsistent quant à l'introduction d'objets ou de relations dans la base de données (cible). Les principales règles sont les suivantes :

- tout objet doit être identifié, recevoir un nom unique, éventuellement un ou plusieurs synonymes et être classé par type,

- pour être valide, un nom d'objet doit entre autres ne pas excéder 30 caractères,
- pour être valides, les noms de synonymes doivent obéir aux mêmes règles que celles qui s'appliquent aux noms d'objets,
- pour tout objet, il existe une liste d'attributs utilisables pour entrer de l'information à propos de cet objet, c'est-à-dire décrire ses propriétés et ses connections avec d'autres objets de la base,
- les types d'objets autorisés et leurs attributs dépendent du méta-modèle contenu dans la méta base de données associée à Quickspec,
- ...

Ces règles clôturant les renseignements ayant trait à Quickspec, nous ne pouvons qu'inviter le lecteur qui serait intéressé par davantage de détails à se reporter à la littérature en la matière, à savoir [MET89b], [MET89c].

En effet, après avoir présenté brièvement MDM et Quickspec, les deux outils qui vont nous permettre d'implanter la méthode SAMPO via l'implantation du méta-modèle que nous avons élaboré à la section 4.1, il est temps, à présent, de passer à la réalisation de cette implantation.

4.2.4 Réalisation de l'implantation

Pour ce faire, l'outil auquel nous allons avoir recours est bien sûr MDM. Comme nous l'avons vu précédemment, son utilisation nécessite le passage par cinq étapes.

Le lecteur attentif aura cependant rapidement remarqué que la première d'entre elles, à savoir l'identification et la formalisation des éléments constitutifs du système à modéliser, a déjà été menée à bien dans la section 4.1. Cette étape n'avait en effet d'autre but que de fournir ce que nous avons baptisé "le méta-modèle de la méthode SAMPO" puisque le système à modéliser est, en ce qui nous concerne, la méthode SAMPO. Tous les commentaires sur cette étape ayant par conséquent déjà été faits, nous n'allons pas revenir ici sur la démarche qui a été utilisée.

Les étapes deux et quatre, par contre, requièrent quelques explications, étant donné le choix qui a dû y être

fait. Ce choix était en fait la conséquence de la possibilité que MDM offrait d'accepter deux types d'approches différentes.

Soit nous pouvions traduire le méta-modèle graphique de SAMPO en langage MDM à l'étape deux et préparer la table XT à l'étape quatre. Cela revenait à suivre l'approche classique présentée au point 4.2.2

Soit nous pouvions utiliser l'étape deux pour définir une version personnalisée d'OPRR et la traduire en langage MDM. L'étape quatre étant alors réservée à la traduction du méta-modèle de SAMPO et à la préparation de la table XT correspondante.

Ne disposant cependant d'aucune documentation et d'aucun exemple sur cette nouvelle approche qui, à priori, pouvait sembler intéressante car plus générale que l'approche classique, nous nous sommes vus dans l'impossibilité de la retenir et nous avons donc opté pour la voie classique.

Les listings obtenus à l'issue des étapes deux et quatre par cette voie, i.e. la traduction en instructions MDM du méta-modèle graphique de SAMPO et la préparation de la table XT correspondante, figurent respectivement en annexes A3 et A4.

Munis de ces deux listings, il ne nous restait plus, conformément aux dispositions prévues dans les étapes trois et cinq, qu'à procéder à leurs compilations respectives au moyen des utilitaires MGEN et XTGEN. Du moins était-ce vrai en théorie, car en pratique, un certain nombre de problèmes se sont posés lors de ces compilations.

Après quelques recherches, nous avons fini par constater que la plupart d'entre eux résultaient du fait que le modèle sur lequel MDM était basé, à savoir OPRR, était en l'occurrence une version sensiblement réduite du modèle OPRR décrit dans le chapitre deux.

Ainsi, un certain nombre de constructions n'étaient-elles pas autorisées comme, par exemple, les propriétés de rôles ou même de types de relations. Les propriétés de types d'objets, quant à elles, ne pouvaient prendre ni la valeur "integer", ni la valeur "real", et l'usage de la valeur "string" était limité à la propriété "basic name". Enfin, seuls les types de relations contenant au moins une connectivité "many" pouvaient être spécifiés, au détriment donc des types de relations dont

les connectivités étaient "one-one/one-one", "zero-one/one-one", "one-one/zero-one" ou "zero-one/zero-one", et qui étaient, pour leur part, relégués au rang des erreurs.

Suite à ces restrictions, nous avons donc été obligés d'ajuster quelque peu la traduction littérale du méta-modèle graphique de SAMPO, afin de la rendre compatible avec les attentes des compilateurs. Ces ajustements expliquent d'ailleurs pourquoi les listings proposés en annexe ne constituent pas une réplique fidèle des schémas élaborés à la section 4.1. : les types de relations possédant des propriétés ont été transformés en types d'objets ; presque toutes les propriétés se sont vues attribuer la valeur "text" ; les connectivités "zero-one" ont été modifiées et sont devenues "zero-many" ; ...

Ces changements, on pouvait s'y attendre, ont bien sûr été à l'origine de pertes d'intégrité, mais ces pertes constituaient le prix à payer pour pouvoir implanter la méthode SAMPO avec Quickspec.

La seconde approche, mentionnée plus haut, et dans laquelle on commençait par définir une version personnalisée d'OPRR, aurait probablement été plus fructueuse. Elle aurait en effet vraisemblablement permis de travailler avec un modèle OPRR plus élaboré et aurait, par conséquent, résolu une série de problèmes. Malheureusement, pour les raisons déjà citées, nous n'avons pu l'adopter.

Arrivés à ce stade, nous avons mené à terme les cinq étapes requises par l'utilisation de MDM et nous disposons enfin d'une base de données contenant le méta-modèle de la méthode SAMPO. Cette base de données possédant, par ailleurs, le même format que la méta base de données de Quickspec, il ne nous suffit plus que d'intervertir les deux pour obtenir le résultat attendu, i.e. une implantation de la méthode SAMPO.

Cette implantation devrait permettre, comme nous l'avons déjà vu, à un utilisateur final du logiciel d'introduire des spécifications SAMPO, de les éditer, de les modifier, ...

C'est ce que se propose de vérifier la section suivante en introduisant un exemple dans le système et en examinant le résultat obtenu suite à cette introduction.

4.3 INTRODUCTION D'UN EXEMPLE

L'exemple auquel nous nous sommes intéressés ici est fictif et a été développé à l'université de Namur [UER85]. Il a trait à l'informatisation d'une firme de vente par correspondance d'articles d'habillement : la firme Petitpas. La réalisation informatique prévue devrait, en résumé, concerner le traitement des commandes des clients, les livraisons à ces clients et les réapprovisionnements chez les fournisseurs.

La raison pour laquelle notre choix s'est porté sur cet exemple vient du fait qu'il a déjà fait l'objet, l'an passé, d'une modélisation selon la méthode SAMPO. Ce lourd et pénible travail n'a donc pas dû être reconduit, puisque nous disposions déjà des résultats obtenus suite à cette analyse [ROB89]. Ces résultats prennent la forme de graphes de discours, de graphes de conversation et d'un ensemble de tables ayant trait, entre autres, aux types de discours, aux actes de langages, aux actes instrumentaux, aux activités, aux positions, ...

Afin de vérifier la validité de cette modélisation, nous sommes donc passés immédiatement à la phase qui nous intéressait, à savoir l'introduction de ces graphes et de ces tables, préalablement convertis en types d'objets, en types de relations, en propriétés, ... dans la base de données cible de Quickspec.

Petitpas étant cependant divisé en quatre types de discours (cf. 1.4.1.1), et la méthode pour introduire les informations relatives à un type de discours étant la même quel que soit le type de discours envisagé, nous nous sommes limités à introduire les deux premiers, à savoir la gestion des commandes clients et la gestion des comptes clients. Ils ont respectivement pour buts (sociaux) de traiter les commandes des clients reçues ainsi que les livraisons (initiales ou différées) et de traiter tout ce qui concerne les paiements effectués par les clients. Le listing reprenant toutes les informations introduites sur ces types de discours figure en annexe A5.

Suite à l'introduction de ces données, nous nous sommes rendus compte que le résultat obtenu possédait, comme c'était prévisible, certaines limites. Celles-ci peuvent s'expliquer,

d'une part, par le fait que nous avons rencontré un ensemble de problèmes inhérents à l'usage même de Quickspec et, d'autre part, par le fait que la méthode SAMPO s'est avérée trop complexe lorsque nous avons tenté de l'utiliser en pratique.

4.3.1 Problèmes inhérents à Quickspec

La principale critique que nous ayons à formuler à l'encontre de Quickspec concerne le très faible nombre de contraintes d'intégrité qu'il vérifie. Le logiciel ne tient compte, en effet, que de deux contraintes : les identifiants et les connectivités maximales.

La première permet d'éviter que l'utilisateur n'introduise deux fois un même nom pour désigner des objets différents, des relations différentes, ... et la deuxième est destinée à l'empêcher d'avoir à un moment donné plus d'une relation entre deux objets si les deux types d'objets correspondants ne sont pas reliés par au moins une connectivité "many".

Ce faible nombre de contraintes vérifiées constitue une lacune importante et grave du logiciel, et ce d'autant plus qu'il ne paraît fournir aucun moyen d'y remédier. Il n'est pas prévu, par exemple, d'interface avec un langage de programmation traditionnel permettant de rédiger, dans ce langage, de petites procédures destinées à s'assurer du respect de contraintes d'intégrité quelconques.

Il apparaît dès lors impossible de faire vérifier à Quickspec des contraintes telles que des exclusions de rôles ou des contraintes plus spécifiques encore :

Citons à titre exemplatif le fait que si une relation "est simultanée avec" existe entre un acte1 et un acte2, la même relation devrait exister entre l'acte2 et l'acte1. Or, dans l'état actuel des choses, cela n'est pas contrôlé et c'est par conséquent à l'utilisateur de veiller à la cohérence des spécifications qu'il introduit.

Un autre exemple concerne les "sous-actes" d'un acte de langage. D'après la méthode SAMPO, tout acte perlocutoire contient un acte illocutoire, qui contient un acte propositionnel, qui, à son tour, contient un acte d'expression. De plus, tout acte de langage est au moins un acte d'expression. Il existe donc une gradation entre ces différents "sous-

actes", obligeant la personne qui modélise selon la méthode SAMPO à considérer tout acte de langage comme étant le plus "riche" de ses "sous-actes". Cette contrainte n'a cependant pas pu être fournie à Quickspec, ce qui est assez gênant.

A côté de ce défaut majeur du logiciel, nous avons également mis à jour quelques uns de ses défauts de jeunesse.

Ainsi avons-nous constaté un problème du côté de l'affichage des relations. Normalement, les relations 1-M sont affichées à l'écran dans de grandes boîtes de dialogue alors que les relations 1-1 sont affichées dans de petites boîtes. Pour certaines relations ternaires cependant, nous avons remarqué que des relations 1-M étaient présentées dans de petites boîtes rendant de la sorte uniquement la première instance visible à l'utilisateur.

Un autre problème relatif à l'affichage concerne les fenêtres. Au début, elles apparaissaient dans le coin supérieur gauche de l'écran, mais, à force d'en ouvrir et d'en fermer, nous nous sommes aperçus que leur affichage avait tendance à se décaler vers la droite et vers le bas. Dans la situation extrême, la fenêtre était tellement décentrée par rapport à l'écran que l'entièreté de son contenu était invisible à l'utilisateur de telle sorte qu'il n'y avait plus moyen d'entrer des données ou même de fermer cette fenêtre. La seule solution consistait alors à fermer la base de données et à l'ouvrir à nouveau juste après.

Cette solution comportait cependant une part de risque puisqu'il est arrivé à plusieurs reprises que Quickspec se bloque en tentant d'effectuer une sauvegarde de la base de données ou même d'une simple relation, entraînant la perte de toutes les données introduites depuis la dernière sauvegarde et obligeant l'utilisateur à relancer le système. Pour se prémunir contre cet état de fait, il est donc nécessaire de sauver régulièrement la base de données sous des noms différents.

4.3.2 Réflexions sur la méthode SAMPO suite à son implantation

Outre les problèmes liés à Quickspec, l'introduction de l'exemple nous a également permis de déceler un certain décalage entre, d'une part, les possibilités qui étaient offertes

par notre implantation et, d'autre part, celles qui ont réellement été utilisées lors de l'introduction de cet exemple.

Ce décalage provient du fait que le méta-modèle que nous avons implanté s'est avéré, en pratique, trop complet pour pouvoir être exploité pleinement. L'ensemble des graphes et des tables résultant de la modélisation de Petitpas [ROB89] n'ont, en effet, pas suffi pour nous permettre de trouver au moins un objet pour chaque type d'objet, une relation pour chaque type de relation et une valeur pour chaque propriété.

C'est ainsi que certains types d'objets de notre méta-modèle sont restés, suite à l'introduction du cas Petitpas, non instanciés. C'est le cas, par exemple, du domaine action, du domaine entité, de la structure d'acte de langage (SACTEL) ou encore de l'acte perlocutoire. D'autres, pour leur part, ne se sont vus attribuer qu'une seule instance tels le discours et l'entité passive.

En ce qui concerne les propriétés, la situation a été identique puisque des propriétés comme la durée de vie d'un acte, la place ou le monde possible d'un acte de langage, et certains composants de la force illocutoire n'ont jamais reçu de valeur.

Pour ce qui est des types de relations, enfin, il est arrivé, parfois, que nous ne disposions que de deux rôles alors que nous voulions introduire une ternaire. Nous étions donc obligés, soit de trouver un troisième rôle, soit de ne pas l'introduire.

Cette absence d'instanciation de certains éléments de notre méta-modèle pourrait être due, bien sûr, au fait que l'exemple traité, i.e. Petitpas, n'était pas assez important ou complet pour pouvoir contenir tous ces concepts. C'est une hypothèse possible. Nous n'en sommes toutefois pas convaincus du tout. Nous croyons plutôt, comme nous l'avons déjà dit, que c'est notre méta-modèle qui contient un nombre trop important d'éléments. Pour être plus exacts encore, étant donné que ce méta-modèle se veut le reflet le plus conforme possible des concepts présents dans la méthode SAMPO, nous croyons que c'est cette méthode qui est basée sur un nombre trop important de concepts, et qui, en outre, en propose des définitions parfois trop confuses.

Pour illustrer cette dernière affirmation, nous ne donnerons qu'un seul exemple, relatif au concept d'acte.

D'après la méthode SAMPO, en effet, les actes sont divisés en deux ensembles disjoints : les actes de langage et les actes instrumentaux. Nous avons cependant trouvé dans la modélisation du cas Petitpas susmentionnée des actes qui ont été considérés à la fois comme actes de langage et comme actes instrumentaux. Cette situation était, en fait, due à une confusion qui avait eu lieu entre, d'une part, des actes de langage et des actes instrumentaux qui sont simultanés (ce qui est autorisé) et, d'autre part, des actes qui sont simultanément actes de langages et actes instrumentaux (ce qui est interdit).

CONCLUSION

Le présent chapitre était articulé autour de trois points : l'élaboration du méta-modèle de la méthode SAMPO, l'implantation de cette méthode au moyen d'un logiciel baptisé Quickspec, et l'examen du résultat obtenu via l'introduction d'un exemple dans le système.

Suite à cet examen, il nous est apparu que le résultat obtenu pourrait, en réalité, être amélioré de trois manières.

Premièrement, au niveau de l'élaboration du méta-modèle, en utilisant un modèle Entité-Relation plutôt qu'OPRR. Ce dernier modèle, bien que plus élaboré en théorie puisqu'il permet de spécifier des propriétés de rôles, souffre en effet de certaines lacunes tant sémantiques que syntaxiques. Rappelons simplement à ce sujet l'absence de notions telles que les contraintes d'intégrité, les relations de généralisation/spécialisation ou encore la place requise par des schémas.

Deuxièmement, au niveau de l'implantation de la méthode, en ayant recours à un outil capable d'enregistrer et de vérifier toutes les contraintes d'intégrité souhaitées, aussi particulières et compliquées soient-elles.

Troisièmement, au niveau de la méthode SAMPO elle-même, en n'en retenant que les concepts essentiels, au détriment de certains concepts dérivés, peu utiles et souvent définis de manière imprécise.

C'est à ces trois améliorations que vont tenter de procéder les deux derniers chapitres de ce mémoire.

CHAPITRE 5

UNE DEFINITION ET UN META-MODELE
MODIFIES POUR SAMPO

INTRODUCTION

Dans le chapitre précédent, nous avons vu que pour pouvoir être exploité pleinement, notre méta-modèle de la méthode SAMPO devait être simplifié. Nous y avons également vu que, dans une certaine mesure, la méthode SAMPO elle-même pouvait faire l'objet de simplifications.

Dans le présent chapitre, nous allons donc tenter de réaliser un certain nombre d'aménagements de la méthode ainsi que de son méta-modèle, en vue de les rendre moins complexes et plus opérationnels. Ensuite, une fois ces aménagements effectués, nous reconstruirons un nouveau méta-modèle en utilisant cette fois le formalisme Entité/Association décrit dans [BOD89], plutôt que le formalisme OPRR.

Notre but n'est pas, ce faisant, de changer de fond en comble, en quelques semaines, le fruit du travail de plusieurs personnes durant plusieurs années. Il consisterait plutôt à faire la part, dans cette méthode, de l'essentiel et de l'accessoire, à identifier les concepts qui font partie du coeur de la méthode et ceux qui ont un caractère plus périphérique.

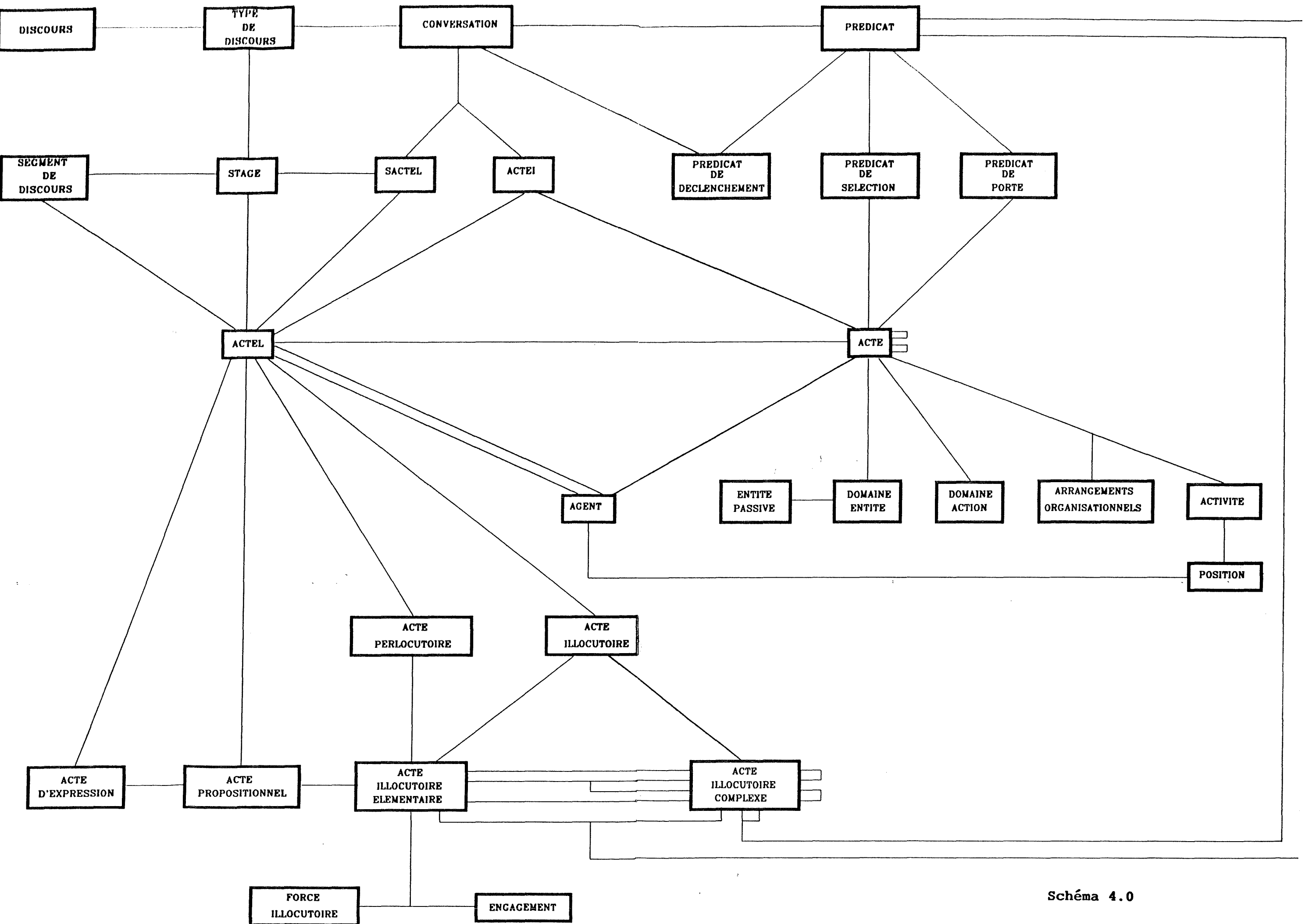


Schéma 4.0

5.1 AMENAGEMENTS DE LA METHODE SAMPO ET DE SON META-MODELE

Pour comprendre le raisonnement qui va suivre, il peut être utile d'avoir en tête une vue globale de la méthode et, pour ce faire, d'avoir sous les yeux le schéma plan 4.0 reproduit ci-contre.

Nous nous proposons, en effet, de passer en revue l'ensemble des éléments représentés sur le méta-modèle élaboré dans le chapitre précédent, autrement dit, l'ensemble des types d'objets, des types de relations et des propriétés présents dans ce schéma plan, en tâchant d'identifier ceux ou celles qui peuvent faire l'objet de suppressions ou de modifications. SAMPO étant, en soi, déjà assez complexe, nous n'avons, en effet, pas cru nécessaire d'encore y faire des ajouts.

Signalons toutefois que le fait de supprimer un type d'objet ne signifie pas pour autant que le concept SAMPO, que ce type d'objet est censé représenter, perd sa définition ou sa place dans la méthode.

Simplement, une telle suppression signifie que le concept SAMPO concerné perd sa représentation dans notre méta-modèle et, par conséquent, la possibilité de faire partie de l'implantation de ce dernier.

5.1.1 Le discours

Ce concept a été défini par SAMPO comme étant "à la fois les actes de langage individuels et les structures d'actes de langage" (cf. 1.4). Il constitue bien sûr un concept important aux yeux de celui qui a conçu la méthode SAMPO puisqu'il représente le plus grand regroupement possible d'actes de langage, l'unité de base qui va faire l'objet d'une analyse. Pour ce concepteur, i.e. Mr. Lyytinen, il existe en effet une infinité de discours potentiellement analysables.

Si on se place au niveau de l'utilisateur du méta-modèle, par contre, c'est-à-dire du spécificateur qui désire analyser un système d'information, parmi cette infinité de discours, il n'y en a plus qu'un seul qui revêt de l'importance : celui qui décrit le fonctionnement du système d'information qu'il a l'intention de spécifier.

La base de données cible de Quickspec ne devrait donc jamais contenir, à un moment donné, la description de plus d'un discours, et le type d'objet DISCOURS ne devrait dès lors jamais se voir attribuer plus qu'une instance à la fois. Il peut, par conséquent, être supprimé sans pour autant léser l'utilisateur. Dans le même temps, il faut bien sûr également supprimer le type de relation DIVISER qui relie DISCOURS à TYPE DE DISCOURS, sans quoi ce type de relation deviendrait caduc puisque ne portant plus que sur un seul type d'objet.

Signalons enfin que l'utilisateur peut toujours, s'il le souhaite, mémoriser le nom du discours qu'il est en train d'analyser en donnant ce nom à la base de données cible dans laquelle il introduit ses spécifications, tout comme on donne à un fichier un nom révélateur de son contenu.

5.1.2 Le domaine entité et le domaine action

Ces deux concepts constituent ce que SAMPO a appelé "l'univers du discours" (cf. 1.3). Tout comme pour le type d'objet DISCOURS, ils peuvent avoir une certaine importance d'un point de vue purement théorique puisqu'ils montrent que le concepteur de la méthode a prévu un nom générique sous lequel il est possible de regrouper l'ensemble des entités ou l'ensemble des actes d'un discours.

D'un point de vue pratique cependant, étant donné que l'utilisateur n'analyse jamais qu'un seul discours à la fois (cf. supra), il n'y aura jamais, à un moment donné, qu'un seul domaine entité et un seul domaine action. On se retrouve donc dans une situation analogue à celle du type d'objet DISCOURS puisque l'on a affaire, ici encore, à deux types d'objets possédant une instance unique.

Les mêmes causes entraînant les mêmes effets, nous avons par conséquent décidé de supprimer ces deux types d'objets et les types de relations y associées. Il s'agit des types de relations : CHANGER qui relie ACTE à DOMAINE ENTITE, INCLURE qui relie ACTE à DOMAINE ACTION et COMPRENDRE qui relie DOMAINE ENTITE à ENTITE PASSIVE.

5.1.3 L'entité passive

L'unique type de relation partant du type d'objet ENTITE PASSIVE ayant été supprimé, ce concept se trouve isolé des autres concepts de la méthode.

La définition que SAMPO lui donne ne fait d'ailleurs que confirmer cet isolement, puisqu'une entité est définie comme "un objet de transaction ou un agent (...) souvent représenté par un nom dans le langage" (cf. 1.3.1).

Autrement dit, les instances de ce type d'objet ne sont que des copies d'instances ou de parties d'instances déjà présentes dans d'autres types d'objets, tels les types d'objets AGENT, ACTE, ... On constate donc une certaine redondance pour obtenir la liste des entités présentes dans un discours.

Or l'utilisateur pourrait aisément se passer de cette redondance si on mettait à sa disposition un langage de requêtes lui permettant de sélectionner seul et de manière dynamique les entités qu'il souhaite parmi les spécifications qu'il a introduites dans sa base de données.

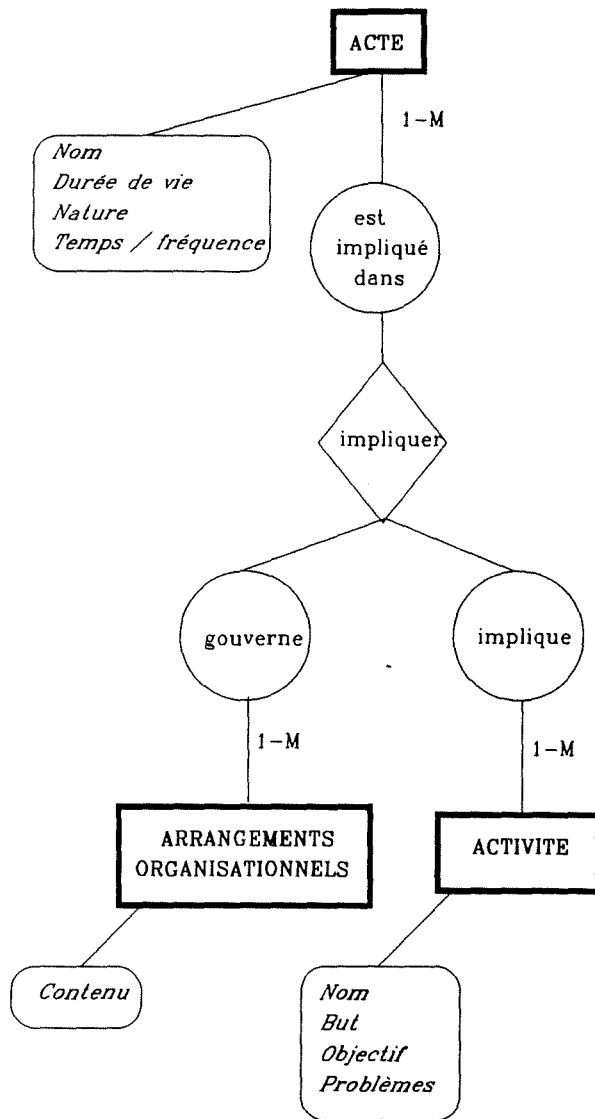
Partant du principe que Quickspec, via les sélections qu'il permet de faire à l'écran, fournit les éléments de base d'un tel langage et que, de toute façon, toute autre implantation devrait être au moins aussi valable dans ce domaine, nous avons décidé de supprimer le type d'objet ENTITE PASSIVE.

5.1.4 Les arrangements organisationnels

Sous le terme arrangements organisationnels, SAMPO reprend tantôt "les contrats qui régissent l'implication d'un acte de langage ou d'un acte instrumental dans une activité" et tantôt "les contrats qui régissent les activités elles-mêmes".

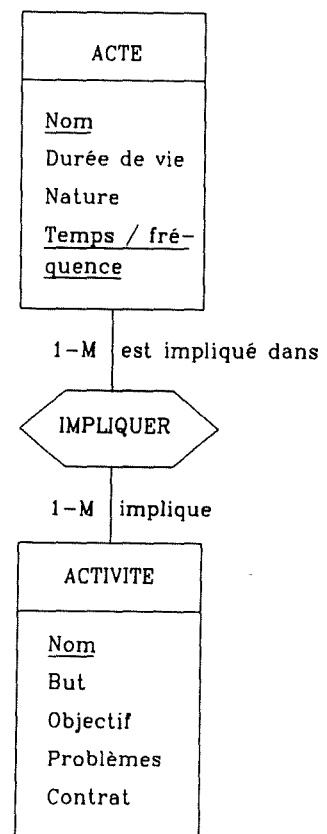
Face à cette alternative, nous avons opté dans un premier temps pour la première définition, comme le confirme d'ailleurs notre méta-modèle.

Après des recherches plus approfondies, nous nous sommes cependant rendu compte que "la notion d'activité permet d'assigner des actes instrumentaux et des actes de langage à des fonctions organisationnelles distinctes" (cf. 1.4.3.1) et



Transformation 5.1

en haut : avant, en OPR
 en bas : après, en E/A



que, par conséquent, la notion d'activité englobe celle d'acte.

Dans un second temps, nous avons donc préféré opter pour la deuxième définition et considérer les arrangements organisationnels non plus comme un type d'objet autonome relié à ACTE et ACTIVITE par une relation ternaire, mais bien comme une propriété du type d'objet ACTIVITE, la relation ternaire devenant par la même occasion une relation binaire (cf. Transformation 5.1 ci-contre).

Ce changement était d'autant plus aisé que le type d'objet ARRANGEMENTS ORGANISATIONNELS ne possédait qu'une seule propriété (contenu) et pouvait donc facilement devenir lui-même propriété d'un autre type d'objet.

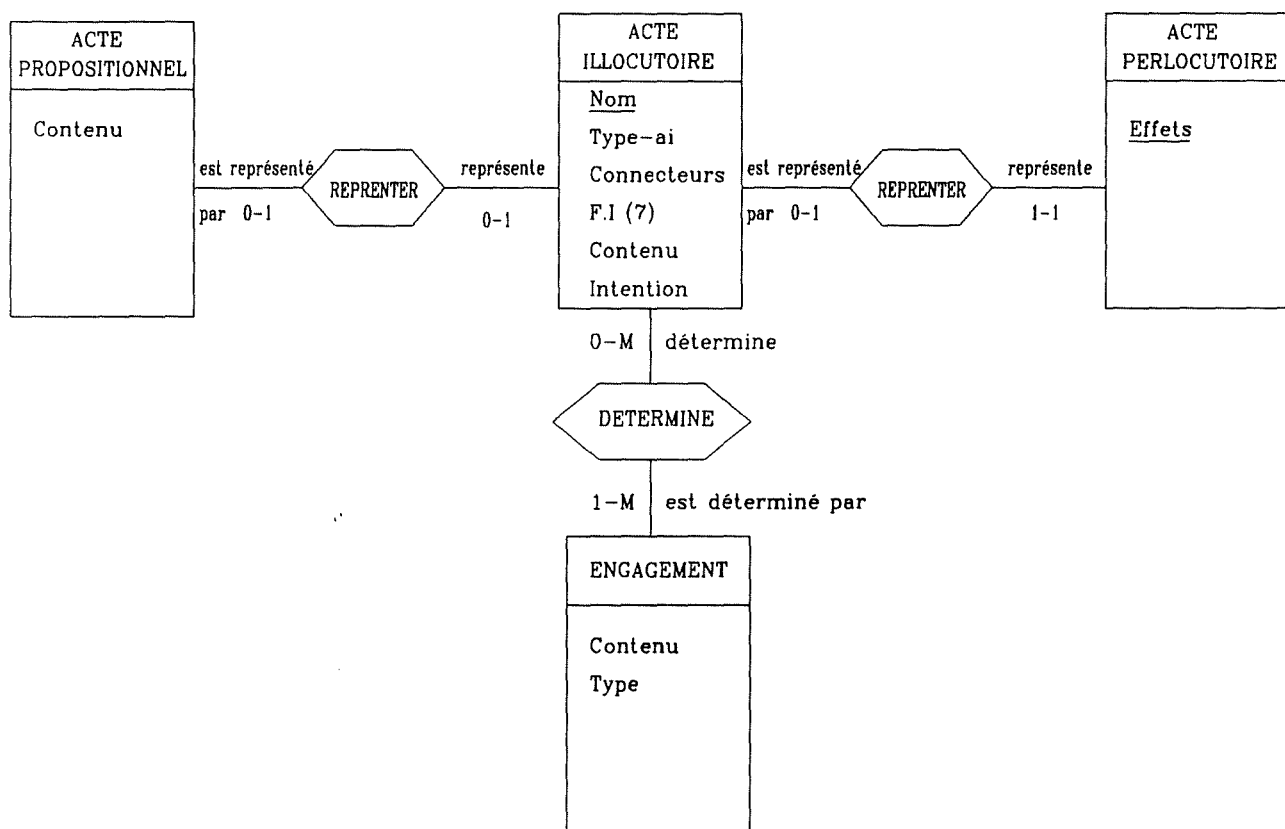
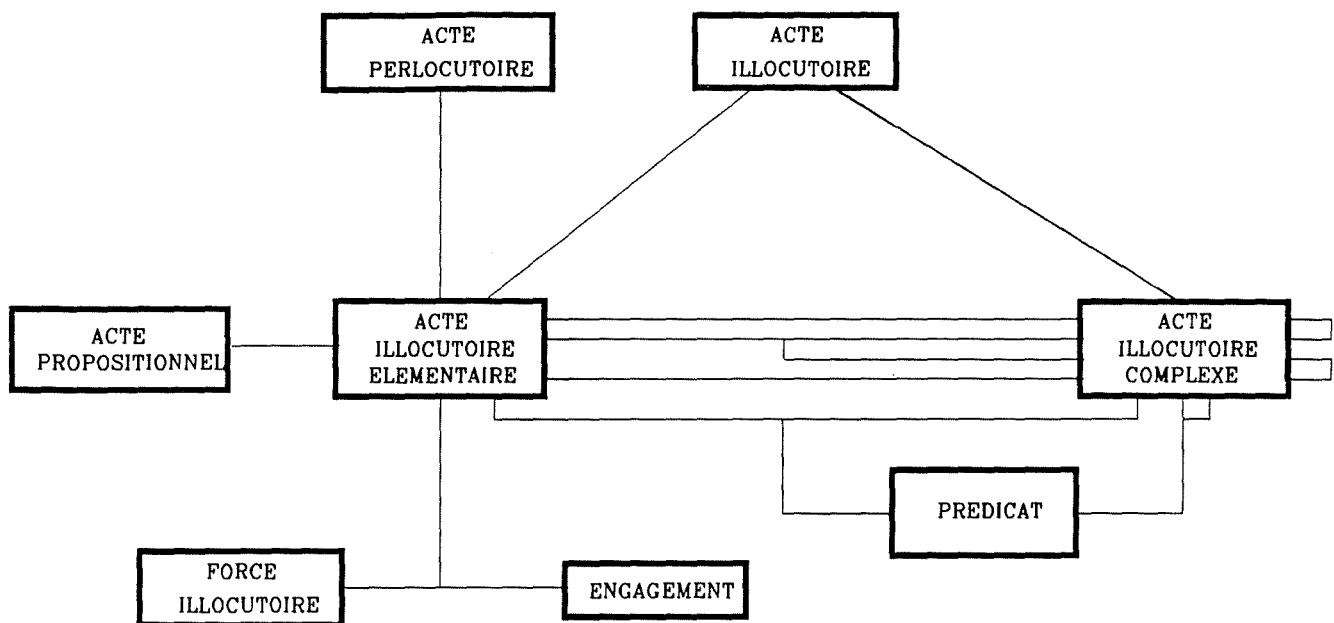
5.1.5 L'acte illocutoire

Comme nous l'avons mentionné au chapitre 1, l'acte illocutoire est à la base de la théorie des actes de langage puisqu'il est considéré comme l'unité de base de toute communication humaine signifiante (cf. 1.3.2.2).

Cette place de choix lui a d'ailleurs valu d'être décomposé de manière plus précise encore, en actes illocutoires élémentaires et actes illocutoires complexes. La distinction entre les deux s'appuie, ainsi que nous l'avons vu (cf. 1.3.2.2), sur les critères suivants :

- l'acte illocutoire complexe se construit récursivement sur l'acte illocutoire élémentaire en utilisant des connecteurs illocutoires (7, & ou =>).
- l'acte illocutoire élémentaire est le seul à posséder un contenu propositionnel et une force illocutoire.
- l'acte illocutoire élémentaire est le seul à pouvoir déterminer des engagements.

Nous n'avons pas l'intention de critiquer ici cette décomposition ou les critères sur lesquels elle s'appuie. Toutefois, nous nous devons de faire remarquer que, bien que fidèle à la théorie, la représentation que nous en avons donnée dans notre méta-modèle ne vérifie en rien que l'acte illo-



Transformation 5.2

en haut : avant, sans propriétés ni noms de types de relations
 en bas : après, avec propriétés et noms de types de relations
 (ou de types d'associations, dans le vocabulaire E/A)

cutoire qu'un utilisateur est en train d'introduire possède une syntaxe correcte.

Cette représentation a en effet un caractère purement descriptif, incompatible avec les attentes que l'on a d'une implantation, comme en témoigne l'exemple suivant, dans lequel on se limitera volontairement à l'introduction d'un acte illocutoire (complexe) sans tenir compte des autres types d'actes de langage.

Mettons-nous un instant à la place de l'utilisateur qui désire introduire l'acte illocutoire suivant, tiré de l'exemple 1.1 : "La chemise n°123 coûte 1299 francs, cravate non comprise". Il va devoir entrer dans sa base de données :

- un acte illocutoire : "prix_chemise_sans_cravate"
- un acte illocutoire complexe : "la chemise n°123 coûte 1299 francs, cravate non comprise"
- deux actes illocutoires élémentaires : "la chemise n°123 coûte 1299 francs" et "cravate non comprise"
- un type de relation COMPOSER entre les deux actes illocutoires élémentaires et l'acte illocutoire complexe

Cela fait ni plus ni moins que cinq informations entrées sans aucune vérification par Quickspec de la cohérence entre ces informations. Suite à ce constat de carence, nous avons opté pour une manière moins descriptive mais plus opérationnelle de représenter les choses. Elle consiste, comme le montre la transformation 5.2 ci-contre, à :

- supprimer les types d'objets ACTE ILLOCUTOIRE ELEMENTAIRE (AIE) et ACTE ILLOCUTOIRE COMPLEXE (AIC) et les remplacer par une propriété du type d'objet ACTE ILLOCUTOIRE (AI) qui indique si celui-ci est élémentaire ou complexe (type_ai)
- supprimer, dans le même temps, tous les types de relations qui reliaient un AIE à un AIC, et les remplacer, dans le cas où un AI est complexe, par une propriété de l'AI qui indique le ou les connecteur(s) illocutoire(s) utilisé(s)
- supprimer les deux types de relations qui reliaient un AIC à un PREDICAT
- supprimer le type d'objet FORCE ILLOCUTOIRE et en faire une propriété de l'AI, pourvu que celui-ci soit élémentaire

- remplacer, par conséquent, la relation ternaire qui reliait AIE, FORCE ILLOCUTOIRE et ENGAGEMENT par une relation binaire qui relie maintenant AI, dans le cas où il est de type élémentaire, à ENGAGEMENT
- remplacer les types de relations REPRESENTER qui reliaient un ACTE PROPOSITIONNEL ou un ACTE PERLOCUTOIRE à un AIE par les mêmes types de relations, mais reliés cette fois à un AI, pourvu que le type de celui-ci soit élémentaire.

Les propriétés associées à un AI seront dès lors les suivantes :

- * type_ai : le type de l'AI (élémentaire ou complexe)
dans le cas où l'AI est de type élémentaire :
 - * intention : l'intention associée à l'AI
 - * force ill. point illocutoire
 - * force ill. mode de réalisation du point illocutoire
 - * force ill. degré de force du point illocutoire
 - * force ill. conditions du contenu propositionnel
 - * force ill. conditions préparatoires
 - * force ill. conditions de sincérité
 - * force ill. degré de force des conditions de sincérité
(cf. définition du type d'objet FORCE ILLOCUTOIRE en annexe A2)
- dans le cas où l'AI est de type complexe :
 - * contenu : le contenu de l'AI
 - * connecteurs : le(s) connecteur(s) illocutoire(s) utilisé(s) (∧, & ou =>)

Suite à ces modifications, il faudra bien sûr ajouter, lors de l'implantation du méta-modèle, une procédure qui sera déclenchée lors de l'introduction d'un AI. Elle aura pour but de faciliter cette introduction et de vérifier de manière dynamique les contraintes d'intégrité que nous venons d'énoncer de manière statique dans le modèle.

Dans ces conditions, l'introduction de l'exemple traité ci-dessus pourrait prendre la forme suivante :

- l'utilisateur introduit dans sa base de données un AI, à savoir "la chemise n°123 coûte 1299 francs, cravate non com-

prise" et précise, via les propriétés de cet acte, qu'il est complexe et construit au moyen du connecteur illocutoire &

- le logiciel prend ensuite la relève et vérifie s'il existe bien, dans la base de données, deux AIE qui sont "la chemise n°123 coûte 1299 francs" et "cravate non comprise". Dans l'affirmative, l'acte introduit est accepté. Dans la négative, il est accepté également, mais uniquement après l'enregistrement automatique par le logiciel des deux AIE manquants et l'introduction, par l'utilisateur, des propriétés relatives à ces deux actes.

L'utilisateur, de la sorte, introduit un minimum de données et bénéficie d'une cohérence maximum entre celles-ci.

5.1.6 La structure d'actes de langage

La théorie des actes de langage, sur laquelle repose la méthode SAMPO accorde, comme son nom le laissait présupposer, une place prépondérante au concept d'acte de langage. Contrairement à son homologue instrumental, l'acte de langage peut être décomposé en sous-actes ou, au contraire, s'unir à d'autres actes pour composer ce que SAMPO a convenu d'appeler une structure d'actes de langage.

Plus exactement, une telle structure se définit comme étant "une séquence ordonnée d'actes de langage, généralement exécutés alternativement par les participants au discours et qui forme un tout logique" (cf. 1.4.1.2).

Le fait de former un "tout logique" ne nous semble cependant pas être une condition suffisante pour justifier la création d'un nouveau concept et pour lui donner un nom distinct.

De plus, ce regroupement au caractère assez arbitraire, étant donné qu'aucun critère précis n'est fourni, ne possède qu'une utilité toute relative, puisqu'il nous a été possible de modéliser deux types de discours du cas Petittpas sans y avoir recours.

Enfin, de par sa présence, ce concept a obligé le concepteur de la méthode SAMPO à définir une conversation comme étant "des *structures d'actes* de langage, des actes instrumentaux et leurs dépendances dynamiques" (cf. 1.4.3.2), ce qui d'après nous constitue une asymétrie regrettable.

Dès lors, pour éviter de s'encombrer d'un concept peu utile et peu opérationnel, et pour pouvoir définir une conversation de manière logique comme étant "des actes de langage, des actes instrumentaux et leurs dépendances dynamiques", nous avons décidé de supprimer le type d'objet STRUCTURE D'ACTES DE LANGAGE. Dans la relation ternaire qui le reliait à CONVERSATION et ACTE INSTRUMENTAL, ce type d'objet a donc été remplacé par le type d'objet ACTE DE LANGAGE. Quant à la relation binaire qui le reliait au type d'objet STAGE, elle a été supprimée.

5.1.7 Le type de relation reliant AGENT à ACTE INSTRUMENTAL

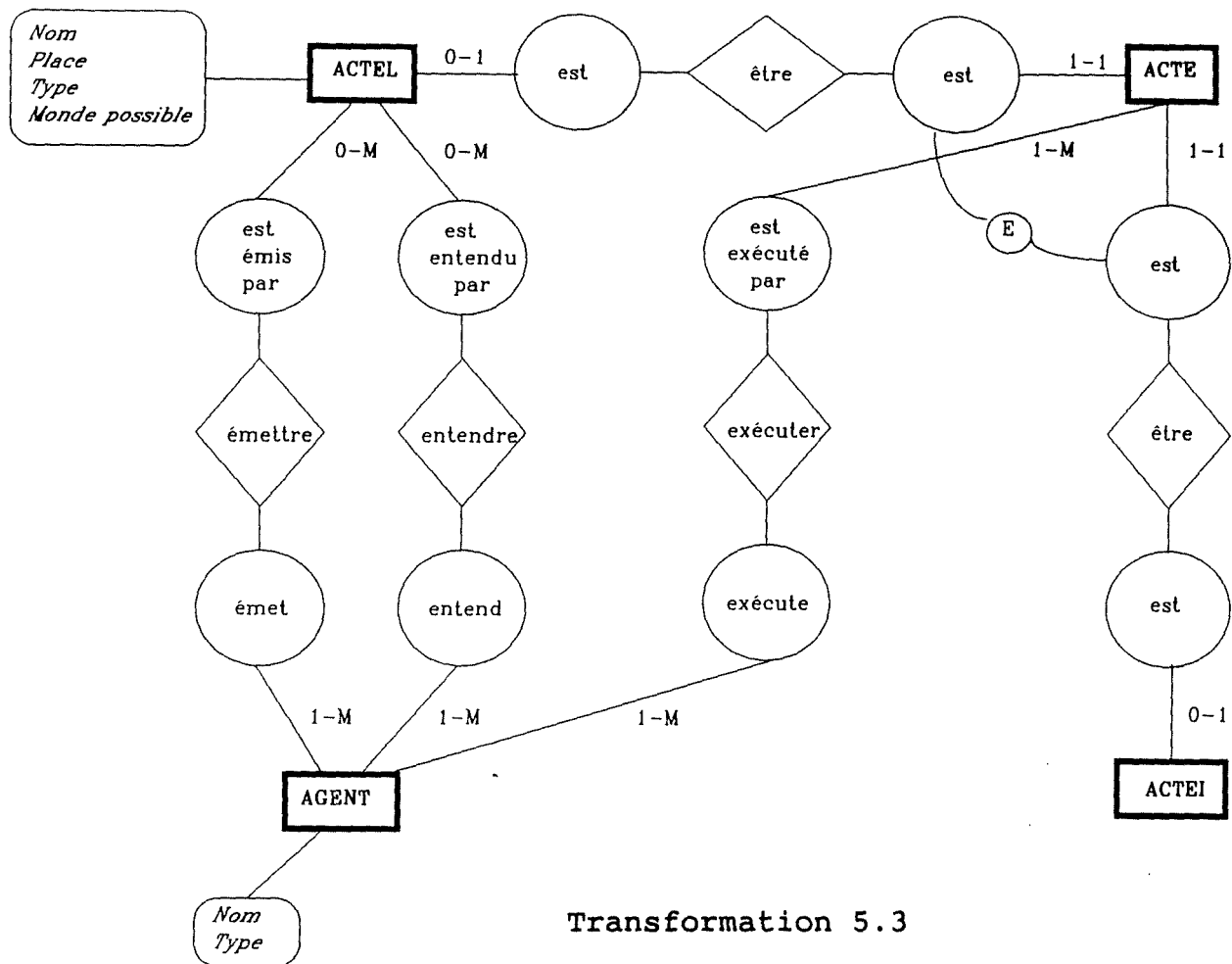
"Un agent est une personne, un groupe de personnes ou une machine qui exécute un (des) acte(s)". "Un acte est un acte de langage ou un acte instrumental". "L'agent qui exécute un acte de langage est appelé locuteur, celui vers qui cet acte est dirigé est appelé auditeur". "L'initiateur d'un acte instrumental est un agent".

Si l'on n'y prend garde, des définitions inter-reliées de ce type peuvent rapidement conduire, lorsqu'on tente de les modéliser, à des lacunes ou, au contraire, à des redondances dans les schémas.

Dans notre méta-modèle, par exemple, lorsqu'on a affaire à un ACTE qui est un ACTE DE LANGAGE, le type de relation qui relie les types d'objets ACTE et AGENT s'avère être redondant avec les types de relations qui relient les types d'objets ACTE DE LANGAGE et AGENT. Lorsque, par contre, on a affaire à un ACTE qui est un ACTE INSTRUMENTAL, ce même type de relation perd son caractère redondant puisque le méta-modèle ne contient pas de type de relation entre les types d'objets ACTE INSTRUMENTAL et AGENT.

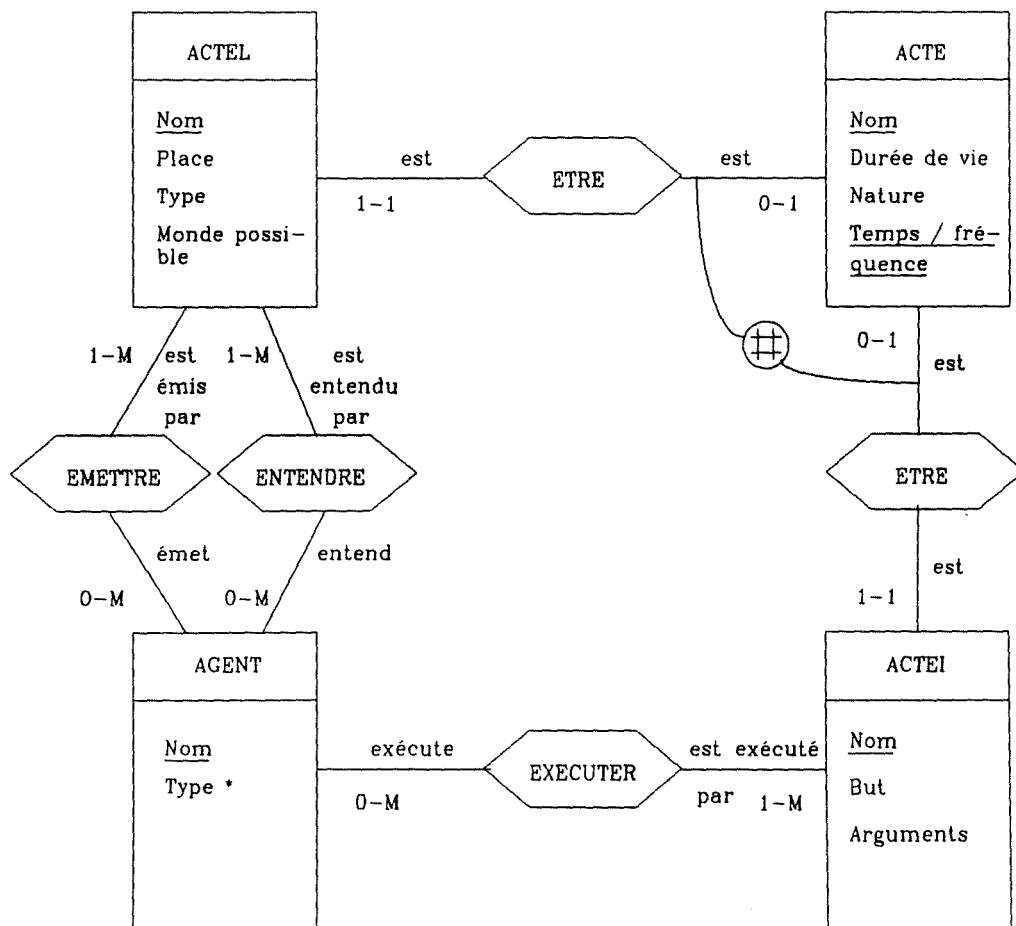
Une erreur de modélisation a donc été commise. Elle peut être corrigée de deux manières, en supprimant chaque fois un des deux éléments redondants.

Une première possibilité consiste donc à supprimer, par exemple, les deux types de relations qui relient ACTE DE LANGAGE et AGENT et à passer, lorsqu'on désire connaître les agents concernés par un acte de langage, par les types de re-



Transformation 5.3

en haut : avant, en OPRR
en bas : après, en E/A



lations existant entre ACTE DE LANGAGE et ACTE puis entre ACTE et AGENT. Cette possibilité n'est cependant pas la meilleure puisqu'on ne sait plus, ce faisant, si un agent est concerné par un acte de langage en tant que locuteur ou en tant qu'auditeur.

Une seconde possibilité, la bonne, consiste par conséquent à garder ces deux types de relations entre ACTE DE LANGAGE et AGENT, et à supprimer le type de relation existant entre ACTE et AGENT, de telle sorte qu'il n'y ait plus de redondance (cf. Transformation 5.3 ci-contre). Dans ce cas cependant, il ne faut pas oublier d'ajouter un nouveau type de relation entre ACTE INSTRUMENTAL et AGENT sans quoi il n'y aurait plus moyen de retrouver l'initiateur d'un acte instrumental à partir de cet acte.

Afin de conférer à la méthode SAMPO un caractère plus opérationnel, comme nous en avons émis le désir dans l'introduction de ce chapitre, nous avons également jugé utile de définir de manière (plus) opérationnelle le concept d'acte ainsi que les "sous-actes" qui entrent dans la composition d'un acte de langage.

5.1.8 Définitions

Jusqu'à présent, l'acte a été défini comme "un acte de langage ou un acte instrumental". L'inconvénient de cette définition réside toutefois dans le fait qu'elle ne permet pas d'identifier un acte de manière directe, sans référence à d'autres concepts. Nous avons donc cherché une définition ayant un caractère plus autonome, et nous avons finalement opté pour la suivante : "Un acte est une partie nécessaire de tout processus de transaction car déterminant son contenu, ce qui doit être échangé. Les actes sont généralement caractéristiques à une organisation".

Par ailleurs, afin de mieux rendre compte de la gradation qui existe entre les différents "sous-actes" d'un acte de langage, nous avons préféré substituer aux anciennes définitions de ces "sous-actes" (cf. annexe A2) les définitions suivantes

ACTE D'EXPRESSION : acte obtenu en émettant des signaux phonétiques ou graphiques.

ACTE PROPOSITIONNEL : acte d'expression ayant un contenu (propositionnel).

ACTE ILLOCUTOIRE : acte d'expression ayant un contenu propositionnel et auquel est associé une intention.

ACTE PERLOCUTOIRE : acte d'expression ayant un contenu propositionnel, auquel est associé une intention et produisant des effets sur le(s) auditeur(s).

Ces quelques définitions clôturent l'ensemble des aménagements que nous avons jugé souhaitables de réaliser sur la méthode SAMPO ainsi que sur son méta-modèle.

Nous pouvons donc à présent, comme cela a été prévu dans l'introduction de ce chapitre, reconstruire un nouveau méta-modèle pour cette méthode.

5.2 RECONSTRUCTION DU NOUVEAU META-MODELE DE SAMPO

La technique utilisée pour ce faire n'a rien de compliqué puisqu'elle consiste à reprendre le méta-modèle du chapitre précédent, à le mettre à jour en fonction des aménagements dont la méthode SAMPO vient de faire l'objet et à traduire le résultat obtenu dans le formalisme Entité/Association.

Pour des raisons mentionnées à plusieurs reprises dans le chapitre précédent (cf. 4.1.2, par exemple) c'est, en effet, ce formalisme qui a été retenu. Il a, rappelons-le, fait l'objet d'une (brève) présentation au chapitre deux (cf. 2.1.3).

Signalons, par ailleurs, que comme nous l'avions déjà fait dans le chapitre 4, nous avons préféré faire précéder les différents schémas par un schéma plan. Celui-ci est présenté sous l'intitulé "schéma 5.0" et est destiné à donner une vue d'ensemble des concepts manipulés par la méthode.

Cet ajout se justifie toutefois moins dans ce cas que pour le premier méta-modèle, étant donné le nombre moindre de types d'objets (ou de types d'entités, dans le vocabulaire E/A) et l'aspect moins étendu des graphiques E/A par rapport aux graphiques OPRR.

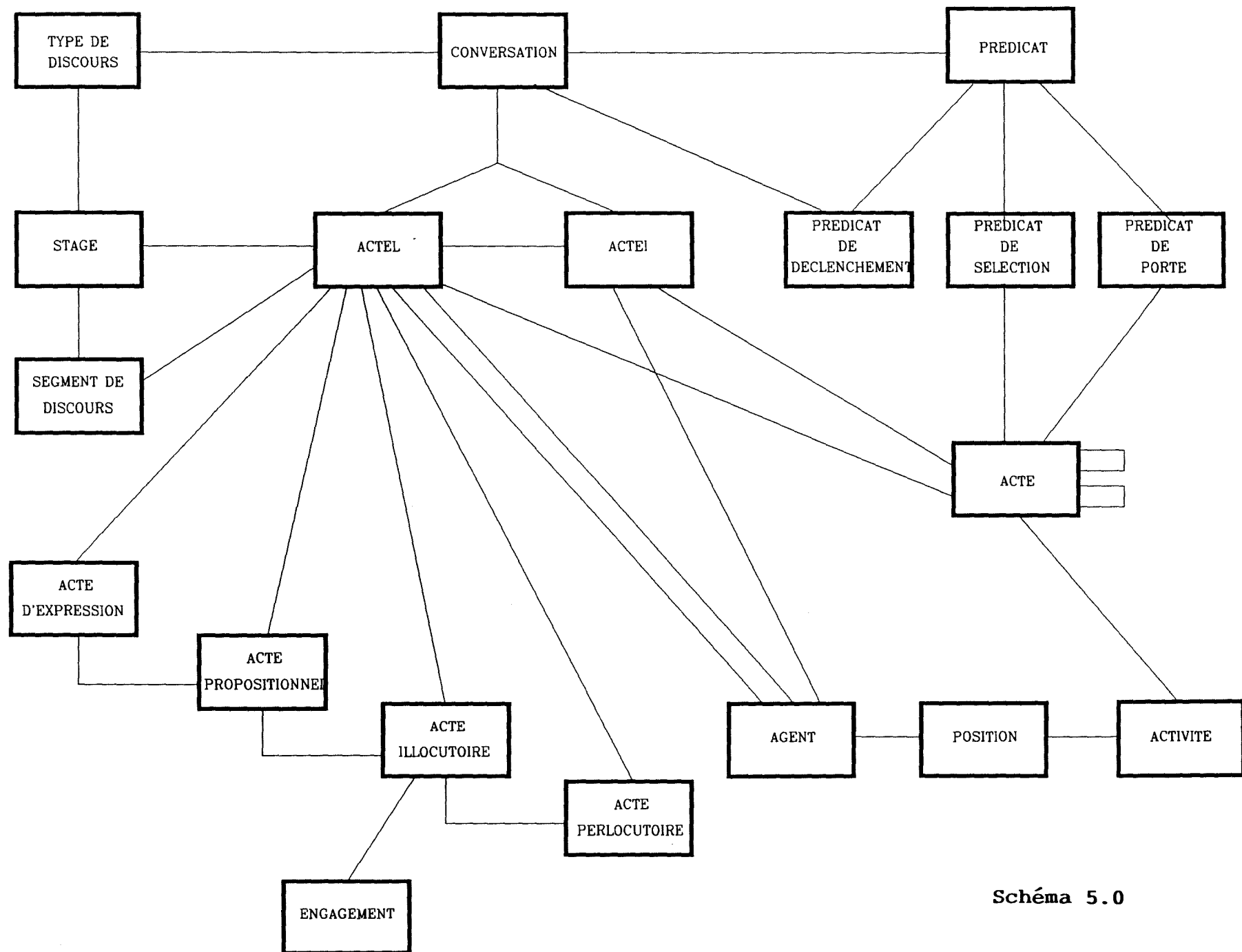


Schéma 5.0

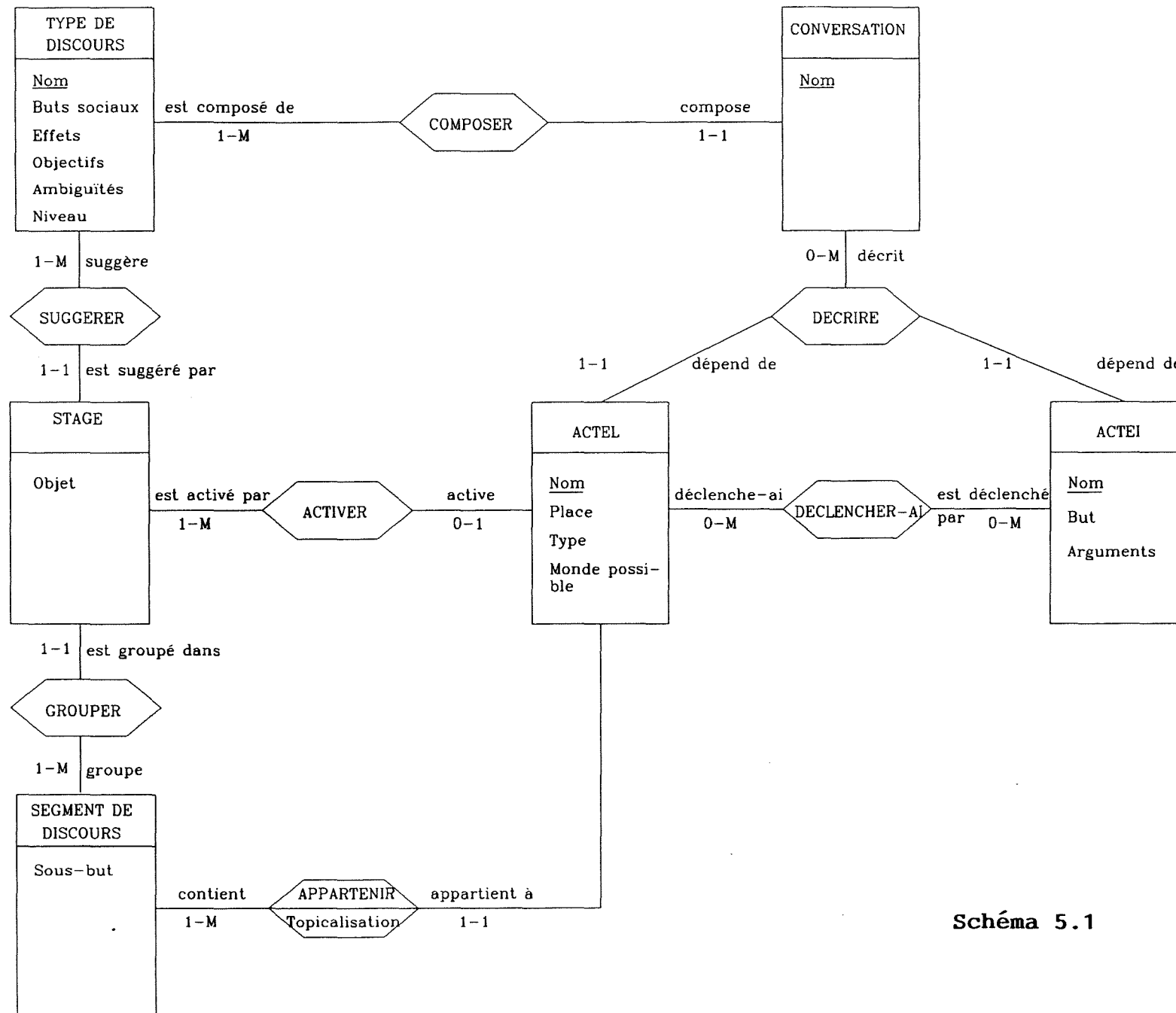


Schéma 5.1

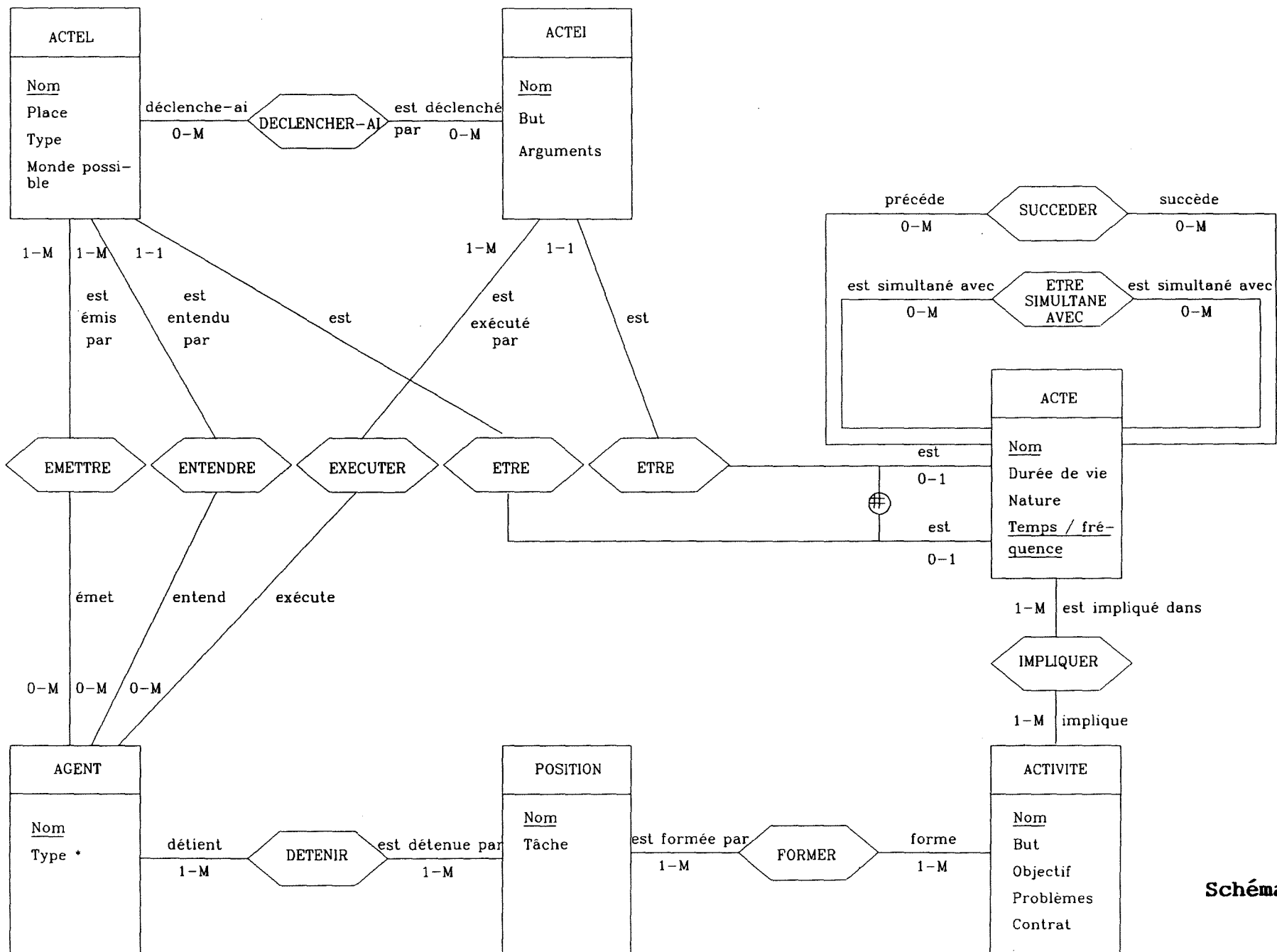


Schéma 5.2

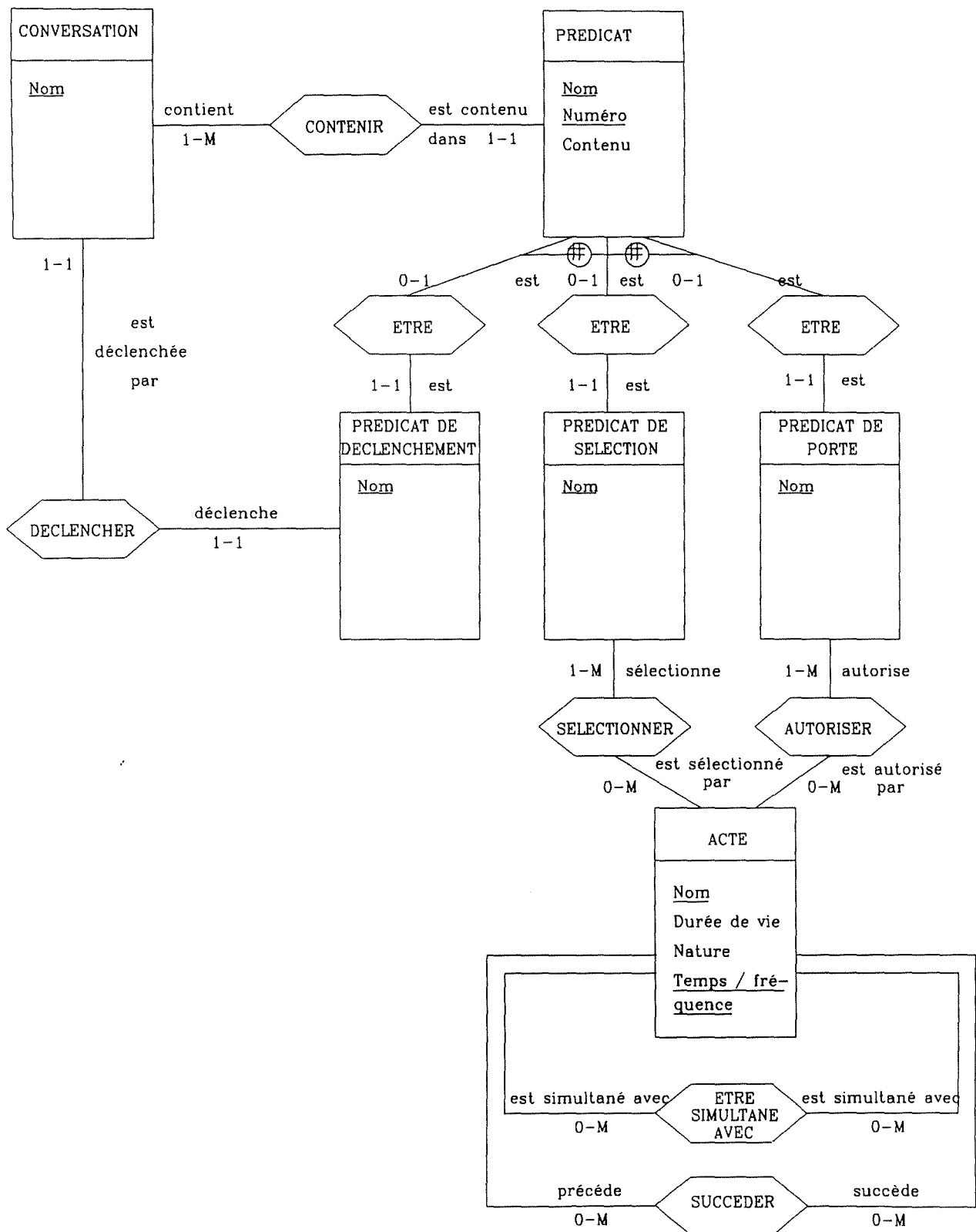


Schéma 5.3

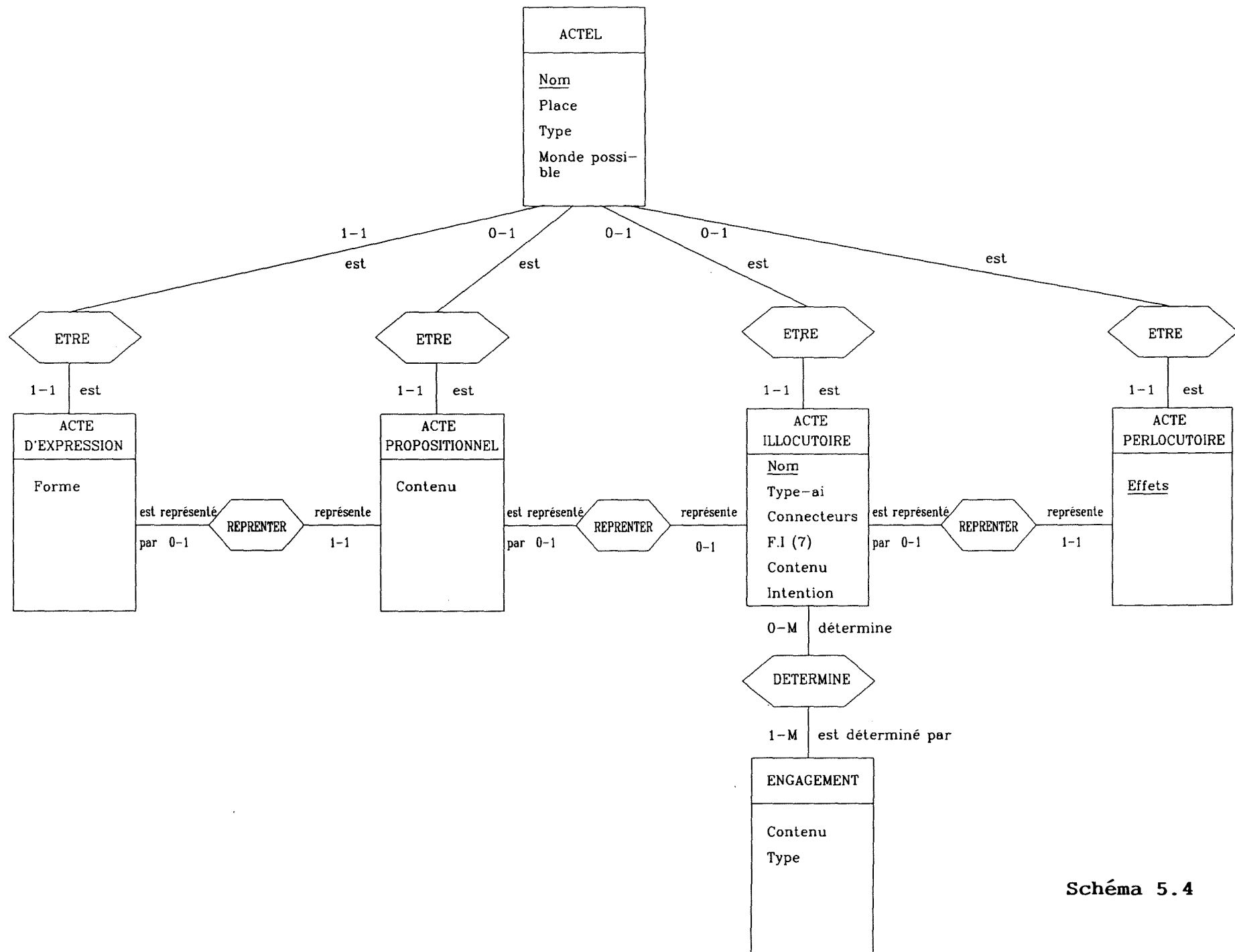


Schéma 5.4

CONTRAINTES D'INTEGRITE.

FI(7) représente, en fait, 7 attributs :

Force ill. point illocutoire

Force ill. mode de réalisation du point illocutoire

Force ill. degré de force du point illocutoire

Force ill. conditions du contenu propositionnel

Force ill. conditions préparatoires

Force ill. conditions de sincérité

Force ill. degré de force des conditions de sincérité

1) Valeur :

- La nature d'un acte ne peut être qu'interne ou externe.
- Le type d'un acte ne peut être que virtuel ou non virtuel.
- Force ill. point illocutoire ne peut être qu'assertif, commissif, directif, déclaratif ou expressif.
- Le type d'un d'engagement ne peut être que de base et/ou secondaire.
- Type-ai d'un acte illocutoire ne peut prendre comme valeur qu'élémentaire ou complexe.
- Connecteurs ne peut prendre des valeurs que dans l'ensemble ($\neg$, $\&$, $\Rightarrow$).

2) Existence

- Dans le cas où Type-ai = "élémentaire" :
l'attribut Intention existe et les attributs FI(7) peuvent exister
- Dans le cas où Type-ai = "complexe" :
les attributs Contenu et Connecteurs existent
- Si un acte de langage est de type "virtuel", alors l'(les) engagement(s) qu'il détermine est (sont) de type "faible".
- Si un agent est de type "machine", il ne peut pas exécuter un acte qui "est" un acte instrumental.

3) Autres :

$AIC ::= AIE$
 $AIC ::= \text{not } (AIC)$
 $AIC ::= AIC \ \& \ AIC$
 $AIC ::= P \implies AIC$

avec AIC : acte illocutoire complexe
 AIE : acte illocutoire élémentaire
 P : prédicat

CONCLUSION

Une méthode SAMPO quelque peu modifiée, quelques changements dans la définition de certains concepts, un nouveau méta-modèle, tels sont les principaux résultats fournis par ce cinquième chapitre.

Nous croyons que, dans une certaine mesure, ces résultats contribuent à rendre cette méthode plus adaptée qu'elle ne l'était à la spécification de systèmes d'information.

Toutefois, nous nous devons de signaler qu'elle ne constitue pas encore, malgré ces améliorations, une référence en la matière et ce, parce que sa principale particularité, à savoir le fait qu'elle est basée sur la théorie des actes de langage, ne la prédispose pas naturellement à ce genre d'application.

Pour justifier cette affirmation, prenons deux exemples parmi d'autres.

Dans une méthode d'analyse du discours, il semble logique qu'un concept tel que l'acte de langage occupe, contrairement à l'acte instrumental, une place prépondérante et intervienne dans de nombreuses relations. Par contre, dans une perspective de spécification de systèmes d'information, cette prédominance de l'acte de langage sur l'acte instrumental ne se justifie plus. Au contraire, il pourrait même parfois être utile, afin de décrire de manière précise certains traitements à effectuer, de pouvoir disposer, pour l'acte instrumental, d'une structure plus complète et de relations plus nombreuses que pour l'acte de langage.

Le second exemple pourrait, pour sa part, avoir trait au concept de prédicat de déclenchement (d'une conversation). Il s'agit à nouveau, en effet, d'un concept qui présente une utilité certaine lorsqu'on spécifie un discours composé de plusieurs conversations, mais qui, une fois que SAMPO est utilisé dans le domaine des systèmes d'information, perd une partie de sa pertinence.

Comme nous l'avons indiqué dans l'introduction de ce chapitre, nous n'avons cependant pas voulu remanier trop en profondeur la méthode SAMPO, et nous n'avons par conséquent pas résolu ce type de problèmes.

Cela étant dit, il nous reste maintenant à voir comment il serait néanmoins possible de réaliser une implantation efficace de cette méthode, i.e. une implantation qui ne posséderait plus les lacunes qui résultaient de l'utilisation de Quickspec.

C'est à ce travail que sera consacré le dernier chapitre de ce mémoire.

CHAPITRE 6

VERS UNE NOUVELLE IMPLANTATION :
EBAUCHE D'UNE SOLUTION

INTRODUCTION

Suite aux aménagements dont la méthode SAMPO a fait l'objet dans le chapitre précédent, il serait judicieux, à présent, d'en fournir une nouvelle implantation.

Cette nouvelle implantation devrait cependant bien évidemment remédier aux lacunes décelées dans la première et dont les principales étaient, le manque de vérification du respect des contraintes d'intégrité, l'impossibilité d'un interfaçage de Quickspec avec un autre langage de programmation, et l'absence d'un langage de requêtes pour obtenir certains résultats non prévus explicitement par le logiciel. Pour résoudre ces problèmes, la nouvelle implantation devrait soit se baser sur un autre CASE toolkit générique que Quickspec, soit ne se baser sur aucun outil CASE et être construite de toutes pièces au moyen d'outils traditionnels.

Ces deux solutions s'avèrent cependant impossibles à réaliser en pratique.

La première parce que nous ne disposons d'aucun autre outil CASE que Quickspec et la seconde parce qu'accomplir un tel travail reviendrait, comme nous l'avons dit dans le chapitre trois (cf. 3.5), à réaliser un CASE toolkit spécifique, ce qui demanderait un temps et des efforts considérables.

Face à cette situation, nous avons dès lors décidé de consacrer ce dernier chapitre, non pas à une nouvelle implantation, mais plutôt à l'ébauche d'une solution.

Par ébauche, nous entendons les grandes décisions qu'il conviendrait de prendre, leurs implications, les grandes lignes d'une solution particulière étant donné certains critères.

Ce chapitre doit donc être considéré plus comme la base d'un développement futur que comme une fin en soi.

6.1 QUELQUES CARACTERISTIQUES INDISPENSABLES DE LA NOUVELLE SOLUTION

1. Dans le chapitre quatre, lorsque nous avons voulu introduire les deux types de discours du cas Petittpas, nous nous sommes basés sur l'ensemble des tables et des graphes de discours et de conversation qui avaient déjà été établis.

Cette démarche n'était cependant pas naturelle. Il faudrait, en effet, idéalement partir du texte français et disposer d'un outil qui génère en sortie ces tables (des types de discours, des actes de langage et instrumentaux, des prédicats, des activités et des positions).

2. On aimerait également générer automatiquement les graphes de discours et de conversation. Ces graphes seraient toutefois plus difficiles à obtenir car leur représentation est principalement basée sur des éléments (actes de langage, instrumentaux, ...) représentés par les symboles graphiques des figures 1.1 et 1.2, qui doivent être disposés de manière intelligente. En effet, il faut, par exemple, représenter des actes de langage successifs l'un à côté de l'autre pour éviter que la flèche représentant cette succession ne coupe d'autres liens du graphe afin que ce dernier reste clair et lisible.

3. Par ailleurs, il serait souhaitable que quand on introduit un objet, une propriété ou une relation, toutes les contraintes d'intégrité relatives à l'élément que l'on vient d'introduire soient vérifiées automatiquement. Quelques contraintes se révèlent être indispensables pour notre problème :

- Les contraintes d'identification :

Si la personne qui spécifie introduit une conversation de nom "Conv", il faudrait que le logiciel vérifie qu'il n'existe aucune autre conversation qui porte ce nom, car le nom d'une conversation est identifiant (cf. schéma 5.1).

- Les contraintes d'exclusion de rôle :

L'enregistrement d'un prédicat de porte devrait être refusé s'il est déjà présent dans la base de données en tant que prédicat de sélection ou prédicat de déclenchement.

- Les contraintes sur le domaine de valeur :

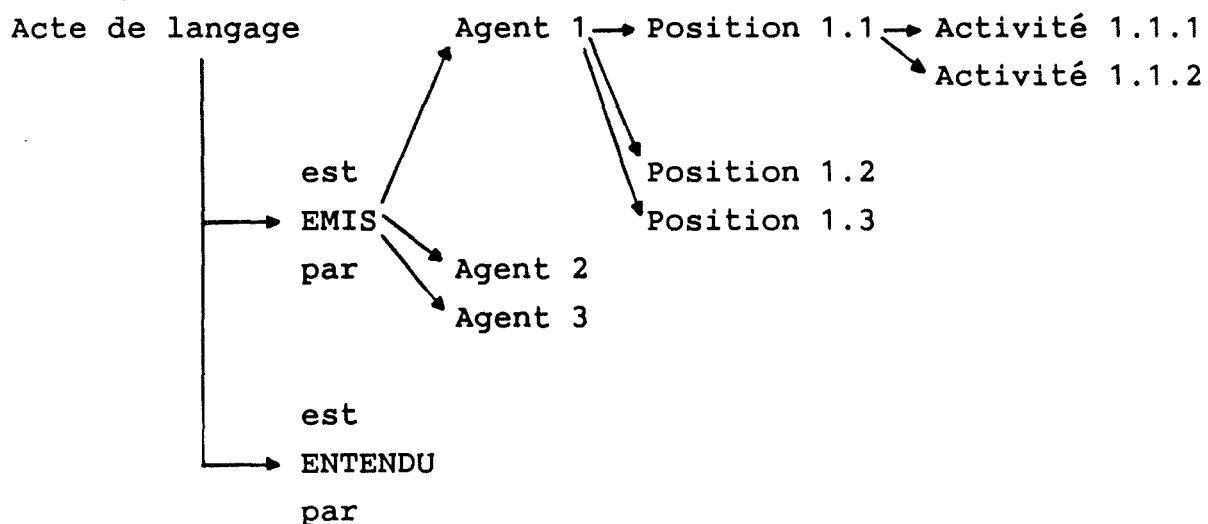
Lorsque l'utilisateur affirme que la nature d'un acte est "XXX", le logiciel devrait signaler que cette propriété de l'acte ne peut prendre que les valeurs "interne" ou "externe".

- Les contraintes de connectivité :

En particulier, un certain automatisme serait souhaitable en matière de vérification des contraintes de connectivité. Lors de l'introduction d'un prédicat de déclenchement, par exemple, le logiciel devrait l'associer à une conversation étant donné le type de relation 1-1/1-1 "DECLENCHER" (cf. schéma 6.1 ci-contre) qui existe entre les deux types d'objets. De même, un acte instrumental ou un acte de langage devrait être associé à un ou plusieurs agents, ces agents à une ou plusieurs positions, chacune de ces positions à, au minimum, une activité (cf. schéma 6.2).

Comme on peut le remarquer, il est très difficile d'activer, c'est-à-dire de parcourir en accord avec les contraintes de connectivité, tous les types de relations, car une telle activation entraînerait rapidement une explosion combinatoire de possibilités.

Exemple :



Il est donc nécessaire :

- soit de limiter les vérifications des contraintes de connectivité au premier niveau. Par exemple : un acte de langage

est EMIS par un (ou plusieurs) agent(s) et un acte de langage est ENTENDU par un (ou plusieurs) agent(s).

- soit de prévoir exactement le chemin à parcourir afin de respecter les contraintes de connectivité. Pour chaque type d'objet, une procédure particulière connaissant ce chemin serait alors déclenchée : au type de discours serait associée la procédure 1, à la conversation la procédure 2, ...
- soit de combiner les deux premières possibilités.

Les première et dernière optiques limiteraient fortement la solution finale, or il s'agit là d'un défaut que nous souhaitons éviter, car cela nous amènerait à reconstruire un logiciel comme Quickspec avec tous les problèmes d'utilisation vus précédemment.

La deuxième optique, quant à elle, est spécifique au problème étudié et ne le résout que partiellement. En effet, si on introduit un type de discours - l'unité la plus grande qui soit - , d'après les contraintes de connectivité des schémas 5.1, 5.2, 5.3 et 5.4, on sera obligé d'introduire tous les agents, toutes les positions, toutes les activités, tous les actes, tous les prédicats, tous les stages et tous les segments de discours qu'il contient ; ce qui est beaucoup trop lourd. Une solution possible à ce problème consisterait à partir non pas du type de discours mais plutôt de l'acte, notion centrale, à partir de laquelle tous les types d'objets, propriétés et types de relations pourraient être instanciés. En effet, contrairement au type de discours qui est composé d'une (ou plusieurs) conversation(s) comprenant de 0 à M acte(s) de langage ou instrumenta(l)(ux), un acte (de langage ou instrumental) ne dépend que d'une et d'une seule conversation entrant dans la composition d'un et d'un seul type de discours. La "centralité" de l'acte ou plutôt le fait que ce soit une notion relativement "élémentaire" permet d'éviter cette explosion combinatoire de possibilités.

En effet, observons les schémas 5.1, 5.2, 5.3 et 5.4 (cf. chapitre 5), et imaginons comment devrait se dérouler l'introduction de spécifications :

L'utilisateur introduit un acte. Le logiciel lui demande une conversation et ses propriétés, à quel type de discours cette conversation appartient, ainsi que les propriétés de ce

type de discours. Il attend également le prédicat de déclenchement de la conversation.

Ensuite, on peut introduire l'éventuel prédicat de sélection ou prédicat de porte associé à l'acte.

Après cela, le programme s'intéresse à la relation "EST" de l'acte. Le choix est offert entre l'acte instrumental et l'acte de langage (l'un ou l'autre mais pas les deux).

Si on se trouve en présence d'un acte instrumental, les agents et les positions qui leur sont associées, ainsi que leurs propriétés respectives devront être introduits.

Sinon, il est nécessaire d'instancier le type de relation "EST" (acte d'expression, acte propositionnel, acte illocutoire ou acte perlocutoire) à sa valeur la plus "élevée" (cf. 4.3.1). Il faut également veiller à ne pas omettre les engagements déterminés par l'acte illocutoire.

Par la suite, le logiciel demandera à deux reprises les agents et les positions qui leur sont associées, ainsi que leurs propriétés respectives : la première est due au type de relation EMETTRE, la deuxième au type de relation ENTENDRE (cf. schéma 5.2).

Si l'acte introduit est un acte de langage, il appartient par ailleurs à un segment de discours. De par le type d'association 0-1 entre ce dernier et le stage, il peut aussi activer un stage dont on précisera l'objet. Le stage, lui-même, appartient au segment de discours sus-mentionné et est suggéré par un type de discours également déjà enregistré. Un acte de langage peut encore déclencher un (ou plusieurs) acte(s) instrumenta(l)(ux). Si c'est le cas, il faudra introduire les données concernant cet (ces) acte(s) et on se retrouvera alors dans la même situation que lorsque l'acte de départ introduit était un acte instrumental. Une procédure récursive sera par conséquent nécessaire pour traiter ce cas.

Enfin, il faudra noter dans quelles activités est impliqué l'acte et, pour chaque activité, quelles positions elle forme.

Schématiquement, le type de chemin qui serait parcouru à partir de l'acte serait dès lors le suivant :

```

* Acte --> Conversation --> Type de discours
                                --> Prédicat de déclenchement
* Acte -?-> Prédicat de sélection
                                -?-> Prédicat de porte
* Acte -?-> Instrumental --> Agent 1 --> Position 1
                                --> ...
                                --> Position m
                                --> ...
                                --> Agent m
                                OU
                                -?-> de langage Expression ?
                                    Propositionnel ?
                                    Illocutoire ? --> Engagement 1
                                                    --> ...
                                                    --> Engagement m
                                    Perlocutoire ?
                                    ---Emettre-- --> Agent 1 --> Position 1
                                                    --> ...
                                                    --> Position m
                                                    --> ...
                                                    --> Agent m
                                    --Entendre-- --> Agent 1 --> Position 1
                                                    --> ...
                                                    --> Position m
                                                    --> ...
                                                    --> Agent m
                                    --> Segment de discours
                                    --> Stage --> Type de discours
                                                --> Segment de discours
                                    -?-> Acte instrumental 1
                                                ...
                                    -?-> Acte instrumental m
* Acte --> Activité 1 --> Position 1
                                --> ...
                                --> Position m
                                --> ...
                                --> Activité m

```

Cette solution, très spécifique il est vrai, constitue à notre sens la seule approche valable susceptible de résoudre le

problème des contraintes de connectivité et, plus généralement, des contraintes d'intégrité.

6.2 DANS QUEL LANGAGE L'IMPLANTER ?

Les procédures qui vérifieront ces contraintes devront être écrites en respectant une certaine syntaxe, un certain langage.

Pour notre problème, il existe principalement deux grandes catégories de langages possibles : les langages traditionnels et les langages à objets.

a) Les langages traditionnels

Pour être retenu, un langage traditionnel devrait posséder quelques caractéristiques minimales : il devrait être d'un assez haut niveau pour permettre une programmation relativement rapide et structurée, être assez efficace pour obtenir une solution optimale, être assez souple pour pouvoir exprimer toutes les contraintes d'intégrité et également faciliter la manipulation intensive de fichiers volumineux, si l'on veut se dispenser de l'emploi d'un SGBD.

b) Les langages à objets

La programmation par objets facilite le développement de systèmes de taille et de complexité croissantes [MAS89]. Les langages à objets permettent, en effet, de manipuler de nombreux types de données puisqu'ils fournissent un style de programmation dirigé par les données, et qui offre une série de qualités dont une importante : l'abstraction de ces données. Cela revient à dire que la modification des structures de données n'entraîne pas de modification des programmes.

En outre, les langages à objets ajoutent à l'abstraction, la notion d'héritage.

Notons qu'on distingue trois familles de langages à objets (les langages de classes, les langages de frames et les langages d'acteurs), qui privilégient chacune un point de vue différent [MAS89]. Pour les langages de classes (point de vue structurel),

l'objet est considéré comme un type de données, un modèle pour la structure de ses représentants physiques.

La notion d'héritage est particulièrement intéressante pour notre problème. Nous sommes, en effet, confrontés à neuf relations "EST", difficilement prises en charge par un SGBD traditionnel (cf. chapitre quatre). Un système de classes et de sous-classes, par contre, pourrait résoudre ce problème.

Une sous-classe est une spécialisation de la description de sa superclasse dont elle hérite. La spécialisation d'une classe peut-être :

- un enrichissement par ajout de nouvelles variables (c'est-à-dire propriétés) ou de nouvelles méthodes (c'est-à-dire procédures).
- la substitution de nouvelles valeurs de variables ou de nouvelles méthodes à celles qui sont normalement héritées.

La figure 6.1 représente les objets Acte et Prédicat. Comme on l'a vu précédemment, un acte est soit un acte instrumental, soit un acte de langage. Ce dernier peut être, en ce qui le concerne, un acte d'expression, un acte propositionnel, un acte illocutoire et/ou un acte perlocutoire. Le prédicat, quant à lui, peut-être un prédicat de déclenchement, un prédicat de sélection, ou un prédicat de porte.

La spécialisation s'opère dans ce cas des deux façons :

- par l'ajout de variables : place, type et monde possible dans l'acte de langage (actel) par rapport à l'acte, par exemple.
- par substitution de nouvelles valeurs de nom de l'acte de langage (actel) ou de l'acte instrumental (actei) par rapport à l'acte.

Exemple :

Classe Acte

Champs Nom

Période de vie

Nature

Temps / Fréquence

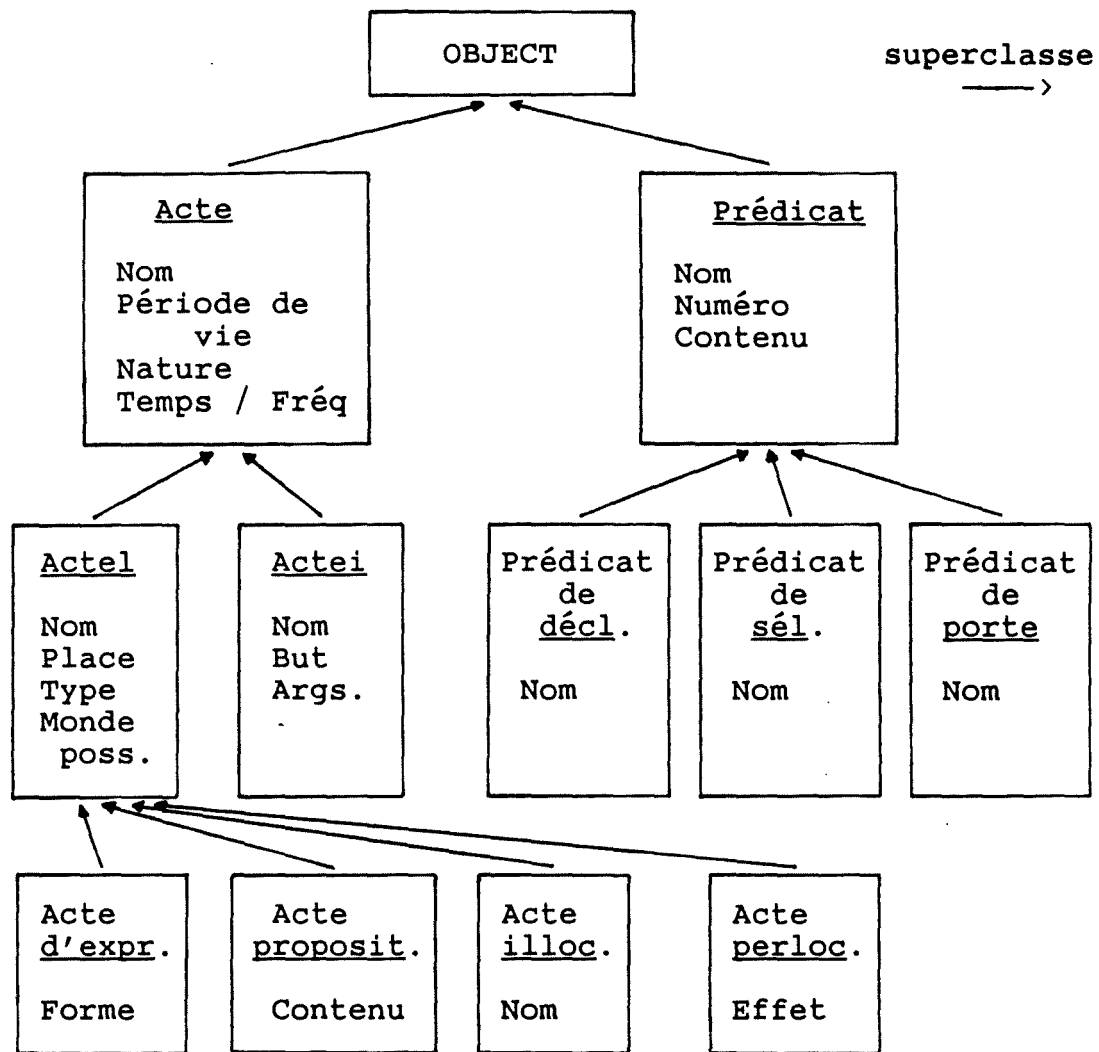


Figure 6.1

Etant donné les possibilités qu'ils offrent, les langages à objets, et plus particulièrement les langages de classes, tel C++ par exemple, nous semblent donc constituer le choix le plus judicieux pour une éventuelle ré-implantation de la méthode SAMPO.

Pour une information plus abondante sur ces langages, le lecteur pourra consulter la littérature [MAS89].

Néanmoins, pour préparer le travail de celui qui souhaiterait implanter SAMPO au moyen d'un langage traditionnel, nous allons, dans les pages qui suivent, voir comment il serait possible de dresser le schéma de la base de données qui servirait de base à cette implantation.

6.3 LE SCHEMA DE LA NOUVELLE BASE DE DONNEES

Comme l'écrit [HAI86], "le concept de base de données se voit associer deux objectifs. Le premier est d'être une représentation fidèle d'un système réel (ce que l'on appelle aujourd'hui « le réel perçu »), essentiellement dans ses aspects statiques. Le second est de constituer un serveur de données opérationnel et efficace pour une gamme de traitements".

Pour construire la base de données (BD), il faut donc partir du schéma conceptuel obtenu en 5.0. Par ailleurs, afin que cette BD soit efficace et opérationnelle, il serait nécessaire également de tenir compte des traitements qui vont aller y puiser des informations.

En ce qui concerne la spécification des structures de données, nous ferons usage d'un modèle spécifique, apte à la fois à représenter la sémantique d'un schéma conceptuel, et à décrire avec précision l'organisation des données du point de vue des accès techniques.

Nous nous inspirerons par ailleurs de la démarche proposée par [HAI86] pour construire le schéma des accès possibles et quelques algorithmes. Comme les traitements dépendent cependant beaucoup des résultats que désire la personne qui implante, nous n'en dresserons pas une liste exhaustive. Il en va de même pour les accès.

6.3.1 Transformation du schéma conceptuel en schéma MAG

Le Modèle d'Accès Généralisé (MAG) est un modèle de spécification de structures de données qui propose un jeu de concepts, à la fois proches de la pratique des fichiers et des BD, et basés sur un modèle rigoureux [HAI86].

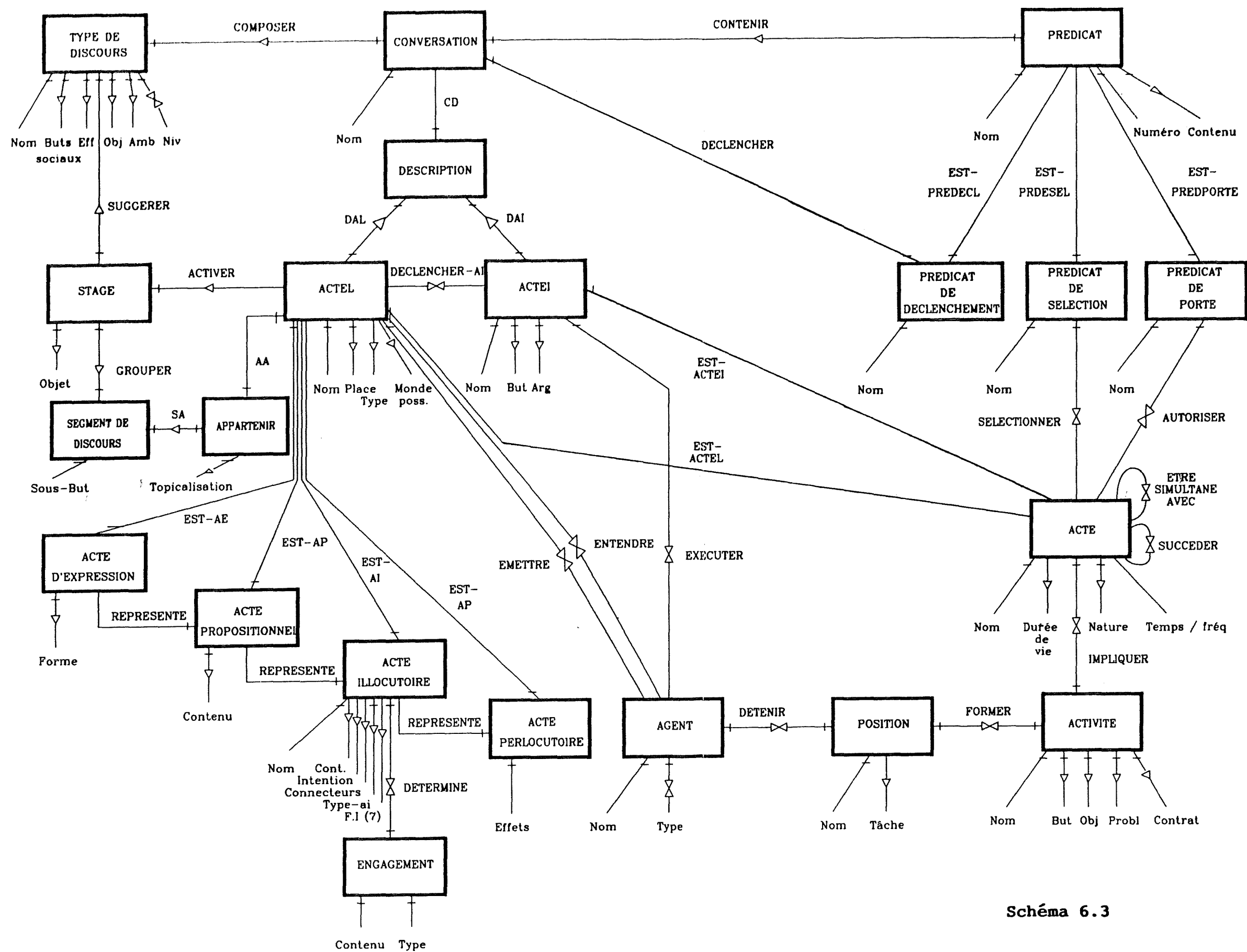
"Cette proximité des outils opérationnels et cette origine en font un outil d'expression adapté à la conversion d'une description conceptuelle en une description exécutable".

Pour le lecteur qui le souhaite, une présentation complète de ce modèle et des objets qui en sont à la base, se trouve dans [HAI86]. Pour notre part, nous le supposerons connu en signalant cependant qu'il permet de spécifier des contraintes d'intégrité.

Nous allons tenter de produire, à partir du schéma Entité/Association (représenté par les schémas 5.1, 5.2, 5.3 et 5.4), un schéma MAG qui soit correct et le plus proche possible du premier. Nous n'utiliserons, à cet effet, qu'une partie du MAG, sans introduire de spécifications concernant les clés d'accès, les types de chemins inverses, ...

La transformation du schéma conceptuel en schéma MAG devrait, par conséquent, se dérouler de manière plus ou moins automatique comme dans [HAI86]. Une entité sera représentée par un article, un type d'entité par un type d'articles, une valeur d'attribut par une valeur d'item, un attribut par un item et un type d'association binaire sans attribut par un type de chemin.

En procédant de la sorte, nous avons obtenu le schéma 6.3.



Notons que le type d'association ternaire DECREIRE (qui figure sur le schéma 5.1) impliquant Conversation, Acte de langage et Acte instrumental, est représenté par le type d'articles DESCRIPTION, et que le type d'association APPARTENIR avec comme propriété Topicalisation, est représenté par le type d'articles APPARTENIR.

Par ailleurs, une fois le schéma MAG construit, il faut encore ajouter les contraintes d'intégrité relatives à ce schéma. Celles-ci ont été rédigées dans un langage proche du langage de désignation de données, tel que l'a défini [HAI86]. "Ce langage permet de désigner des sous-ensembles de données (valeurs d'items et articles). Il nous permet d'exprimer des contraintes d'intégrité et sert de base au langage d'expression d'algorithmes LDA (cf. supra). Ce langage de désignation s'appuie sur une description des données correspondant au noyau sémantique du MAG. Dans ce noyau, on ne retient d'un item et d'un type de chemins que l'association entre deux ensembles de données, sans référence au sens de l'accès qui y est spécifié". Le lecteur pourra se référer à [HAI86] pour disposer de la syntaxe de ce langage.

Les contraintes d'intégrité se présentent comme suit :

- ```

1) - Pour tout p de PREDICAT :
 PREDICAT-DE-DECLenchement (EST-PREDECL : p) OU (1)
 PREDICAT-DE-SELECTION (EST-PREDSEL : p) OU
 PREDICAT-DE-PORTE (EST-PREDPORTE : p)

- Pour tout a de ACTE :
 ACTEI (EST-ACTEI : a) OU
 ACTEL (EST-ACTEL : a)

2) - Nature (: ACTE) = "Interne" OU "Externe"
- Type (: ACTEL) = "Virtuel" OU "Non virtuel"
- Type (: ENGAGEMENT) = "Faible" OU "Non faible"
- Niveau (: TYPE-DE-DISOURS) = "Base" ET/OU "Secondaire"
- Type (: ACTE-ILLOCUTOIRE) = "Elémentaire" OU "Complexe"
- Connecteur (: ACTE-ILLOCUTOIRE) = "&", "&" ET/OU "=>"
- Force-ill-point-ill (: ACTE-ILLOCUTOIRE) = "Assertif"
 OU "Commissif"
 OU "Directif"

```

(1) OU = "ou" exclusif

OU "Déclaratif"

OU "Expressif"

```

3) - Si Type (: ACTE-ILLOCUTOIRE) = "ELEMENTAIRE"
 alors il existe Intention (: ACTE-ILLOCUTOIRE)
 Force-ill-point-ill (: ACTE-ILLOCUTOIRE)
 Force-ill-degré-force (: ACTE-ILLOCUTOIRE)
 il peut exister
 Force-ill-réal-pt-ill (: ACTE-ILLOCUTOIRE)
 Force-ill-cond-cont-prop
 (: ACTE-ILLOCUTOIRE)
 Force-ill-cond-prépar (: ACTE-ILLOCUTOIRE)
 Force-ill-cond-sincér (: ACTE-ILLOCUTOIRE)
 Force-ill-dg-force-cond-sincér
 (: ACTE-ILLOCUTOIRE)

 Sinon il existe Contenu (: ACTE-ILLOCUTOIRE)
 Connecteur (: ACTE-ILLOCUTOIRE)
- Si Type (: ACTEL) = "Virtuel"
 alors Type (: ENGAGEMENT (DETERMINE : ACTE-ILLOCUTOIRE
 (EST-AI : ACTEL))) = "Faible"
- Si Type (: AGENT) = "Machine"
 alors il n'existe pas ACTEI (EXECUTER : AGENT)

```

### 6.3.2 Quelques requêtes intéressantes

Rappelons que la solution idéale devrait posséder trois caractéristiques qu'on n'obtenait pas avec Quickspec :

- 1) La génération automatique des tables de l'analyse du discours
- 2) La production automatique de graphes de discours et de conversation
- 3) La vérification automatique des contraintes d'intégrité

Pour atteindre ces objectifs, il est possible d'élaborer dès à présent et à titre exemplatif, quelques algorithmes qui pourraient figurer dans la nouvelle implantation. Construire tous les algorithmes serait certes intéressant mais prendrait un temps considérable.

Les algorithmes suivants ont été rédigés dans le langage LDA décrit dans [HAI86]. "LDA (Langage de Description d'Algorithmes) tente de réaliser l'intégration de primitives d'accès aux données dans des structures algorithmiques tradi-

tionnelles". "De même que l'existence du MAG se justifie par la nécessité d'exprimer des structures de données indépendamment de celles des SGD, celle d'un langage d'expression d'algorithmes est rendue nécessaire dans une démarche de conception des traitements qui se veut générale et donc indépendante des langages de programmation effectifs. Si les "pseudo-langages", ainsi qu'on les désigne souvent, sont largement répandus, ils pèchent en particulier en ce qui concerne l'accès aux données à structure complexe et leur gestion".

1) La génération automatique des tables de l'analyse du discours

Il est relativement facile de générer les tables de l'analyse du discours. Si l'on désire, par exemple, obtenir la table des types de discours, il suffit de parcourir la base de données et d'imprimer tous les articles Type de discours ainsi que leurs propriétés. On obtient, pour ce faire, l'algorithme suivant :

```
for D := DISCOURS do
 for TD := TYPE-DE-DISCOURS do
 print Nom (: TD);
 print Buts-sociaux (: TD);
 print Effets (: TD);
 print Objectifs (: TD);
 print Ambiguïtés (: TD);
 print Niveau (: TD);
 endfor;
endfor;
```

Algorithme 6.1

La démarche serait très semblable pour les autres tables de l'analyse du discours. Ainsi, pour obtenir la table des actes de langage, il suffira, en plus, de chercher dans quelle(s) activité(s) est impliqué l'acte de langage, son (ses) locuteur(s) et son (ses) auditeur(s).

Pour disposer de tous les locuteurs associés à un acte de langage de nom : "nomactel", on peut utiliser l'algorithme :

```
for D := DISCOURS do
 for AG := AGENT (EMETTRE : ACTEL (: Nom = "Nomactel"))
 do
 print Nom (: AG);
 endfor;
 endfor;
```

### Algorithme 6.2

En combinant ces deux algorithmes, il sera possible de retrouver toutes les tables de l'analyse du discours.

#### 2) La production automatique de graphes de discours et de conversation

Comme, en outre, il a été écrit dans le chapitre 1 (cf. 1.4.3.1) que les éléments d'un graphe de discours étaient :

- un ensemble d'activités
- un ensemble d'actes
- (un ensemble d'entités) (1)
- un ensemble de prédicats
- un ensemble d'actes illocutoires
- un ensemble de relations entre ces objets,

il devrait être possible de reconstituer un graphe de ce type avec les symboles de la figure 1.1 et des algorithmes du genre 6.1 et 6.2, qui se basent sur le contenu de la base de données.

Chaque type de discours trouvé grâce à l'algorithme 6.1 fera l'objet d'un graphe de discours.

Une démarche identique pourrait d'ailleurs être adoptée pour élaborer les graphes de conversation.

#### 3) La vérification automatique des contraintes d'intégrité

Pour la vérification des contraintes d'intégrité, il faut vérifier qu'une conversation introduite, de nom "Conv" par exemple, ne soit pas déjà dans la base avec un nom identique.

---

(1) Supprimé dans le chapitre 5



```

for D := DISCOURS do
 for CO := CONVERSATION (: Nom = "Conv") do
 print " La conversation existe déjà";
 if-no CO then
 Enregistrement-de la-conversation;
 endfor;
endfor;

```

### Algorithme 6.3

Pour vérifier les contraintes d'exclusion de rôles, la requête serait semblable à l'algorithme 6.3.

Les contraintes sur le domaine de valeur ne nécessiteraient pas de requêtes sur la base de données avant l'enregistrement de l'instance dans cette BD. Mais, il faut quand même vérifier, avec des procédures du type 6.4, que la valeur que prend la propriété se trouve dans le domaine de valeurs autorisé. Quand on introduit un acte, par exemple, la vérification de la nature de l'acte s'accomplira comme suit :

```

input (Xnom);
input (Xduréedevie);
input (Xnature);
input (Xtempsfréq);
if Xnature <> "interne" ou "externe"
then
 print "La nature de l'acte ne peut être qu'interne ou
 externe";
endif;

```

### Algorithme 6.4

Pour les contraintes de connectivité, on sera amené à exécuter une combinaison des requêtes vues précédemment, car il sera nécessaire de naviguer dans la base de données (cf. supra).

#### 6.3.3 Simplification du schéma MAG

En observant le schéma 6.3, on peut constater qu'à plusieurs reprises apparaissent des types de chemins "EST". Bien que nécessaires, conceptuellement parlant (cf. 4.1.2), ces types de chemins de généralisation/spécialisation ne doivent pas pour autant être implantés tels quels.

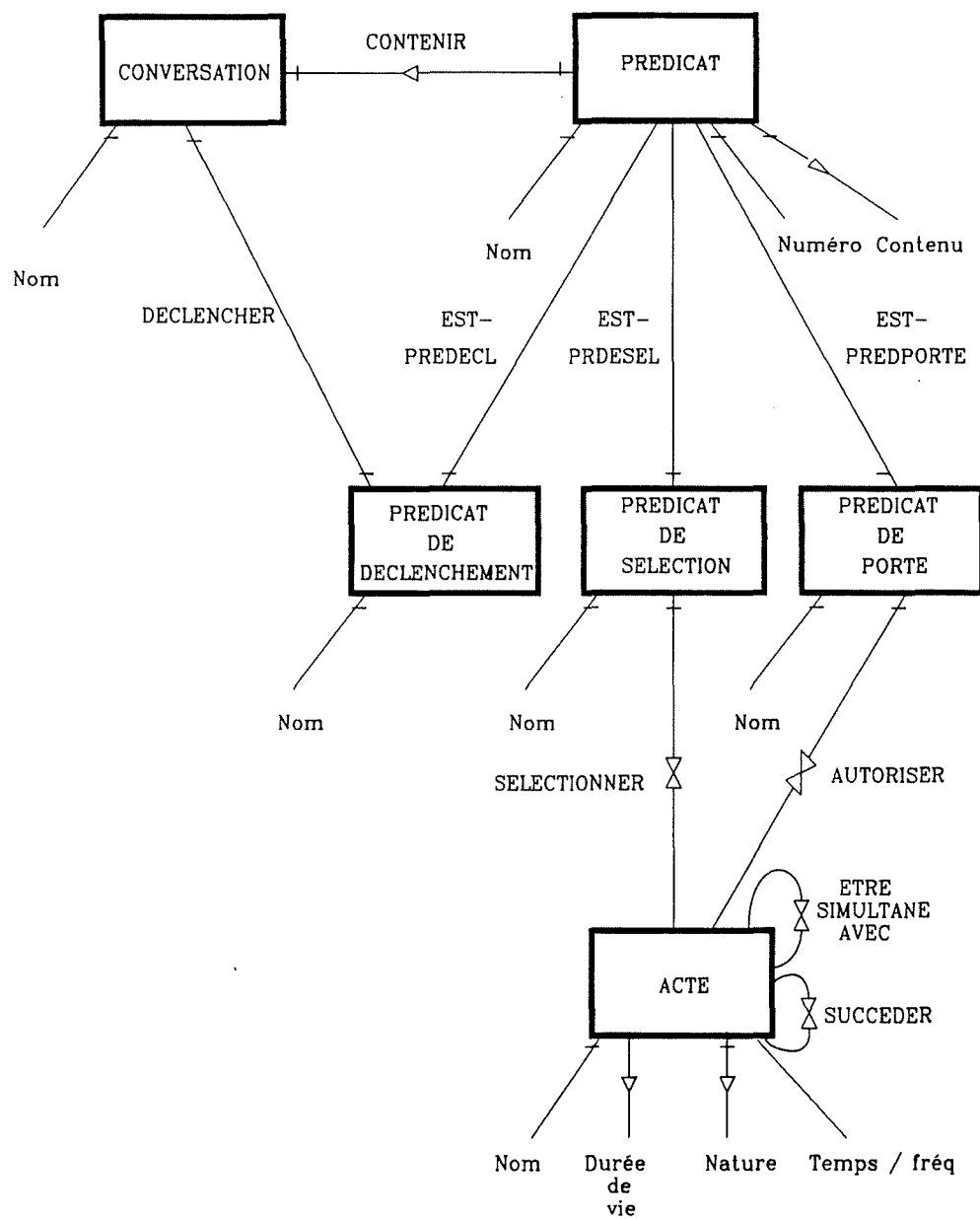


Schéma 6.4a

Ainsi, par exemple, dans le cas du type d'articles PREDICAT (cf. schéma 6.4a), il est tout à fait imaginable de les remplacer par une propriété "Type" qui prendrait comme valeur : prédicat de déclenchement, prédicat de sélection, ou prédicat de porte. Les trois types d'articles correspondants seraient par conséquent supprimés, le type de chemins entre CONVERSATION et PREDICAT-DE-DECLenchement déplacé entre CONVERSATION et PREDICAT et les types de chemins SELECTIONNER et AUTORISER associeraient désormais PREDICAT et ACTE.

Ces transformations, ainsi que les contraintes supplémentaires qu'elles engendrent, peuvent se représenter comme suit :

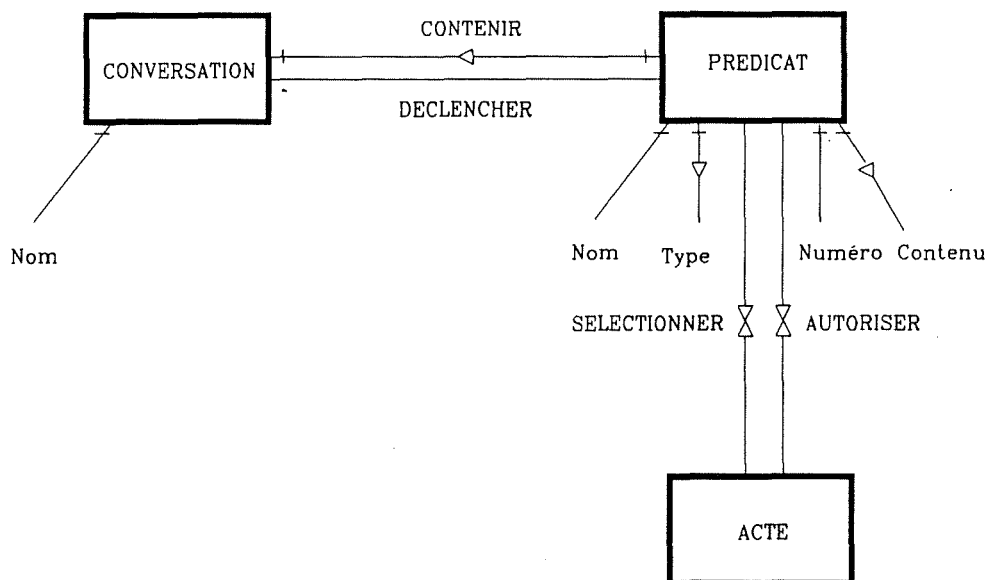


Schéma 6.4b

Contraintes supplémentaires :

- Type ( : PREDICAT) = "Prédicat de déclenchement" OU  
 "Prédicat de sélection" OU  
 "Prédicat de porte"
- Il existe PREDICAT (SELECTIONNER : ACTE)  
 si Type ( : PREDICAT) = "Prédicat de sélection"
- Il existe PREDICAT (AUTORISER : ACTE)  
 si Type ( : PREDICAT) = "Prédicat de porte"

Une démarche analogue peut être menée à bien pour l'acte de langage issu du schéma 6.3 (dont nous avons la partie concernée en 6.5a). Elle consiste, dans ce cas, à supprimer l'acte

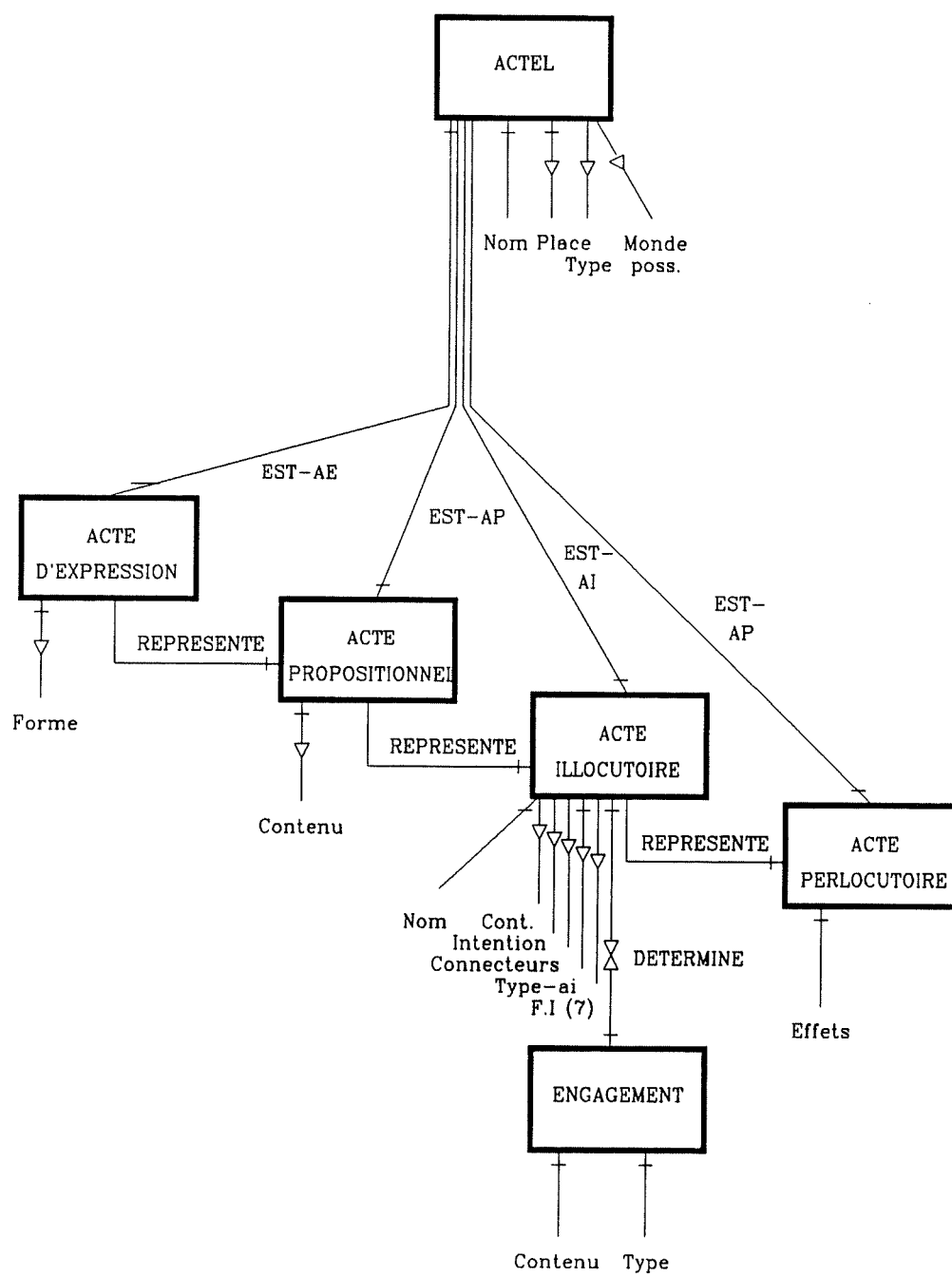


Schéma 6.5a

d'expression, l'acte propositionnel, l'acte illocutoire et l'acte perlocutoire et à ajouter, en contrepartie, une propriété "Sorte-actel" qui prendra comme valeur la plus contraignante des quatre valeurs évoquées ci-dessus. Rappelons qu'un acte perlocutoire est aussi un acte illocutoire qui est à son tour un acte propositionnel ; l'acte propositionnel étant également un acte d'expression.

Le type de chemins DETERMINE est dès lors déplacé entre ENGAGEMENT et ACTEL. Ce type de chemins existera uniquement lorsque "Sorte-actel" vaudra "Acte illocutoire" ou "Acte perlocutoire" (car un acte perlocutoire est également un acte illocutoire).

Toutes les propriétés perdues suite à la suppression des types d'articles sont reportées vers ACTEL.

Parmi ces propriétés (Forme, Contenu, Effets, Intention, Connecteurs, Type-ai et les sept composants de la force illocutoire), seule la propriété "Forme" est obligatoire, indépendamment de "Sorte-actel", car tout acte de langage est au moins un acte d'expression et donc possède une forme.

Les autres propriétés seront obligatoires ou non selon la valeur de "Sorte-actel" (cf. infra).

Le schéma 6.5b illustre les modifications effectuées.

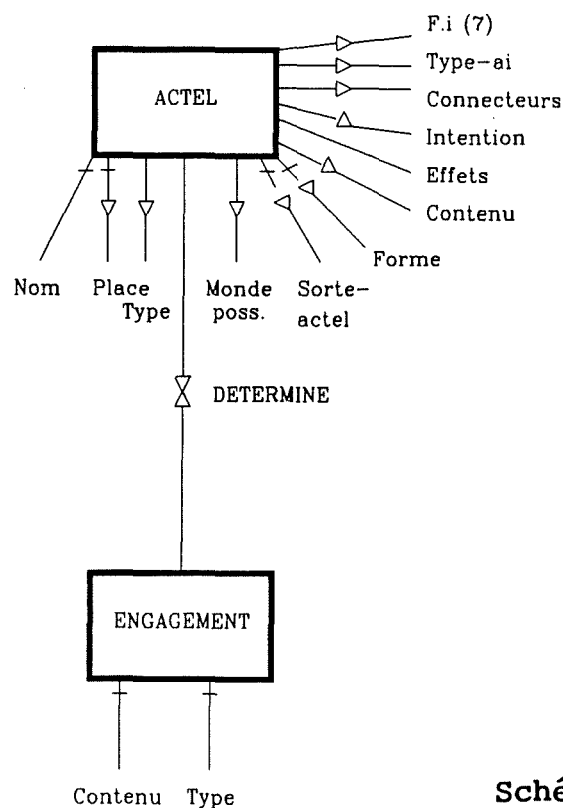


Schéma 6.5b

## Contraintes supplémentaires :

- Sorte-actel ( : ACTEL) = "Acte d'expression" OU  
                                   "Acte propositionnel" OU  
                                   "Acte illocutoire" OU  
                                   "Acte perlocutoire"  
     avec Acte propositionnel -est aussi-> Acte d'expression  
           Acte illocutoire      -est aussi-> Acte propositionnel  
           Acte perlocutoire     -est aussi-> Acte illocutoire
- Il existe ACTEL (DETERMINE : ENGAGEMENT)  
   si Sorte-actel ( : ACTEL) = "Acte illocutoire" OU  
                                   "Acte perlocutoire"
- Forme ( : ACTEL) : obligatoire car tout ACTEL est au moins un acte d'expression
- Contenu ( : ACTEL) : obligatoire  
   si Sorte-actel ( : ACTEL) = "Acte propositionnel" OU  
                                   "Acte illocutoire" OU  
                                   "Acte perlocutoire"
- Effets ( : ACTEL) : obligatoire  
   si Sorte-actel ( : ACTEL) = "Acte perlocutoire"
- Type-ai ( : ACTEL) : obligatoire  
   si Sorte-actel ( : ACTEL) = "Acte illocutoire" OU  
                                   "Acte perlocutoire"
- Connecteurs ( : ACTEL) : obligatoire  
   si \* Sorte-actel ( : ACTEL) = "Acte illocutoire" OU  
                                   "Acte perlocutoire"
- et  
     \* type-ai ( : ACTEL) = "Complexe"
- Intention ( : ACTEL) et 5 composantes de la force illocutoire (cf. schéma 6.1) obligatoires  
   si \* Sorte-actel ( : ACTEL) = "Acte illocutoire" OU  
                                   "Acte perlocutoire"
- et  
     \* type-ai ( : ACTEL) = "Elémentaire"

En examinant le schéma 6.3 (après les transformations établies en 6.4b et 6.5b), on constate qu'il ne reste plus, à présent, que deux types de chemins "EST" : "EST-actei" et "EST-actel". Une fois encore, il est possible de supprimer les deux

types d'articles associés et de les remplacer par un attribut "Sorte-acte" qui pourra prendre les valeurs "Actel" OU "Actei".

Suite à la disparition de ces deux types d'articles, il faut cependant également supprimer le type d'articles DESCRIPTION. Ce dernier provenait, rappelons-le, de la relation ternaire entre CONVERSATION, ACTEL et ACTEI (cf. schéma 5.1) qui, en fait, pouvait se décomposer en deux relations binaires. En conséquence, on crée, entre CONVERSATION et ACTE, le type de chemins DECRIRE et on déplace vers ACTE les types de chemins ACTIVER (qui se trouvait entre STAGE et ACTEL), AA (entre APPARTENIR et ACTEL), EMETTRE et ENTENDRE (entre AGENT et ACTEL), EXECUTER (entre AGENT et ACTEI) et DETERMINE (entre ENGAGEMENT et ACTEL).

Les types de chemins AA, EMETTRE et ENTENDRE sont obligatoires lorsque l'acte concerné est un acte de langage (Sorte-acte = "Actel"). Le type de chemins EXECUTER est obligatoire quand on est en présence d'un acte instrumental (Sorte-acte = "Actei").

Le type de chemins DETERMINE est obligatoire quand l'acte est un acte illocutoire ou perlocutoire (Sorte-actel = "Acte illocutoire" ou "Acte perlocutoire" et, par conséquent, Sorte-acte = "Actel").

Les propriétés seront obligatoires ou non selon "Sorte-acte" et "Sorte-actel" (cf. infra). Il faudra tenir compte également de la valeur de Type-ai (le type d'acte illocutoire : élémentaire ou complexe) pour les propriétés Connecteurs et Intention.

Le schéma 6.6b illustre les changements réalisés.

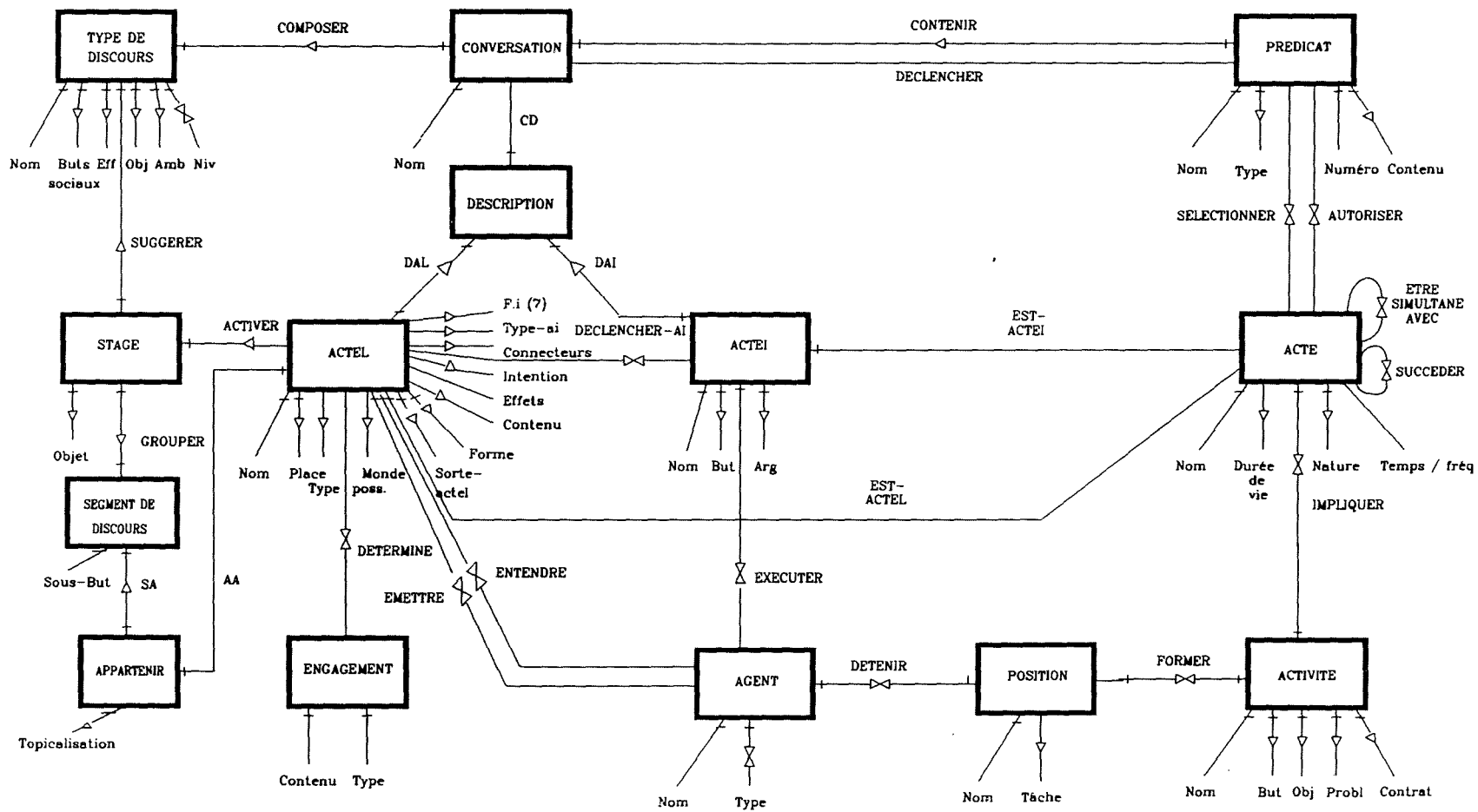


Schéma 6.6a



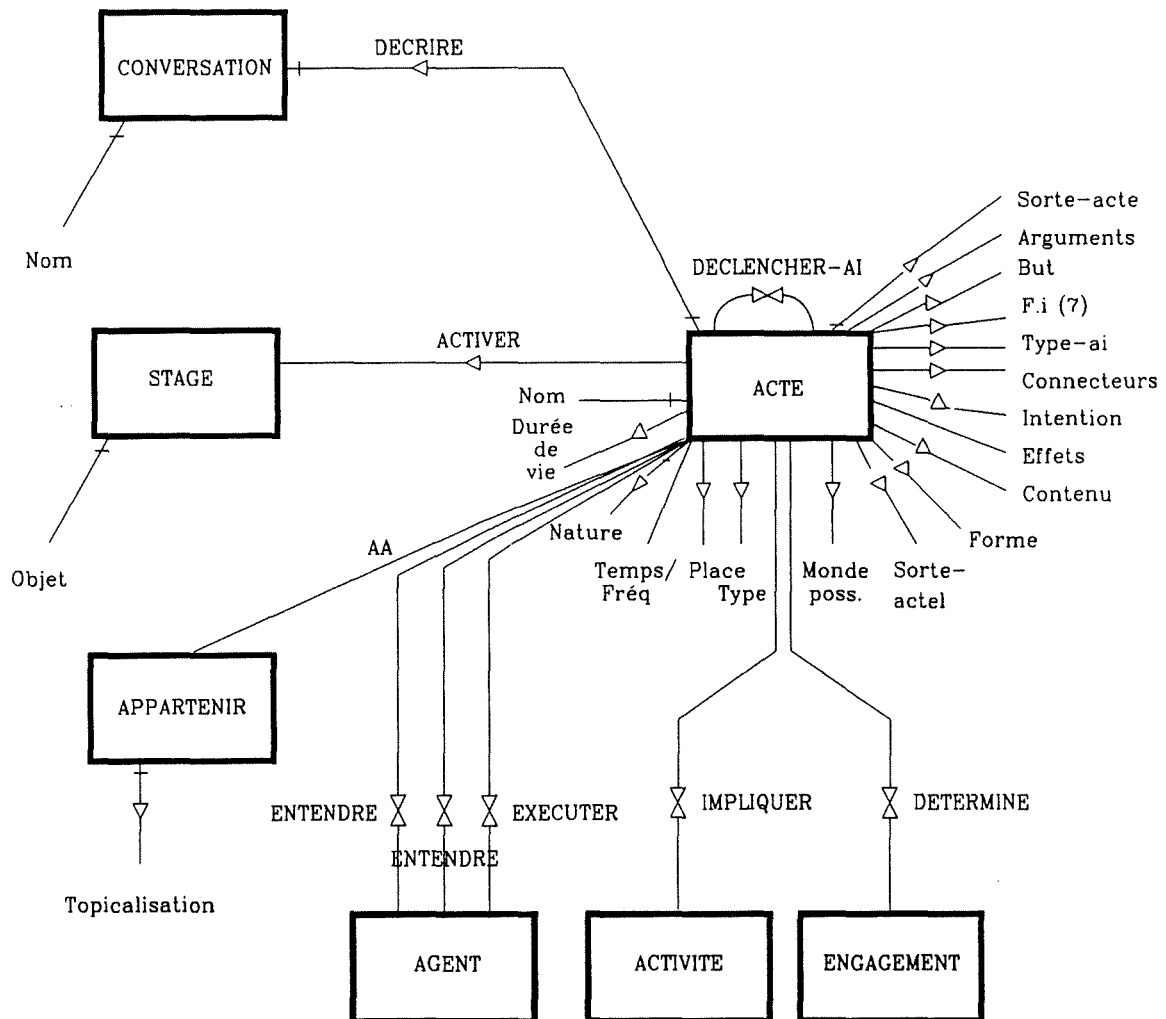


Schéma 6.6b

## Contraintes supplémentaires :

- Sorte-acte ( : ACTE) = "Actel" OU "Actei"
- Il peut exister ACTE ( ACTIVER : STAGE)  
  si sorte-acte ( : ACTE) = "Actel"
- Il existe ACTE ( AA : APPARTENIR)  
                  ( EMETTRE : AGENT)  
                  ( ENTENDRE : AGENT)  
  si sorte-acte ( : ACTE) = "Actel"
- Il existe ACTE ( EXECUTER : AGENT)  
  si sorte-acte ( : ACTE) = "Actei"
- Il existe ACTE ( DETERMINE : ENGAGEMENT)  
  si sorte-acte ( : ACTE) = "Actel"  
  et sorte-actel ( : ACTE) = "Acte illocutoire" OU  
                              "Acte perlocutoire"
- Il existe Place ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"
- Il peut exister Type ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"
- Il existe Sorte-actel ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"
- Il peut exister Monde-possible ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"
- Il existe Forme ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"
- Il existe Contenu ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"  
  et sorte-actel ( : ACTE) = "Acte propositionnel" OU  
                              "Acte illocutoire" OU  
                              "Acte perlocutoire"
- Il existe Effets ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"  
  et sorte-actel ( : ACTE) = "Acte perlocutoire"
- Il existe Type-ai ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"  
  et sorte-actel ( : ACTE) = "Acte illocutoire" OU  
                              "Acte perlocutoire"
- Il existe Connecteurs ( : ACTE)  
  si sorte-acte ( : ACTE) = "Actel"  
  et sorte-actel ( : ACTE) = "Acte illocutoire" OU  
                              "Acte perlocutoire"

```
et type-ai (: ACTE) = "Complexe"
- Il existe Intention (: ACTE) et 5 composants de la
 force illocutoire (cf. schéma 6.1)
 si sorte-acte (: ACTE) = "Actel"
 et sorte-actel (: ACTE) = "Acte illocutoire" OU
 "Acte perlocutoire"
 et type-ai (: ACTE) = "Elémentaire"
- Il existe But (: ACTE)
 si sorte-acte (: ACTE) = "Actei"
- Il existe Argument (: ACTE)
 si sorte-acte (: ACTE) = "Actei"
- Acte-1 déclenche-ai Acte-2
 si sorte-acte (: Acte-1) = "Actel"
 et sorte-acte (: Acte-2) = "Actei"
```

#### 6.3.4 Transformation du schéma MAG (après simplification)

Une fois le schéma MAG élaboré, [HAI86] construit le schéma des accès nécessaires, à partir des accès utilisés par les algorithmes effectifs. Comme nous n'avons cependant donné qu'un petit échantillon d'algorithmes possibles, nous ne pouvons pas dresser la synthèse de tous ces accès. Celle-ci devrait dès lors être établie lors de la nouvelle implantation effective.

## CONCLUSION

## CONCLUSION

Après trois chapitres destinés à présenter les éléments dont nous avons besoin pour atteindre notre but, nous avons, dans le quatrième chapitre de ce mémoire, modélisé la méthode SAMPO au moyen du modèle OPRR (Object-Property-Role-Relationship) et implanté cette méthode en utilisant un CASE toolkit générique dont le nom est Quickspec.

Comme nous l'avons déjà mentionné dans la conclusion de ce chapitre, ces deux réalisations n'ont cependant atteint, qu'en partie, le résultat escompté et ce, en résumé, pour les raisons suivantes :

- SAMPO est une méthode trop complexe et peu opérationnelle,
- OPRR, bien qu'étant assez élaboré, souffre de certaines lacunes tant sémantiques que syntaxiques,
- Quickspec ne vérifie que très peu de contraintes d'intégrité.

Dès lors, forts de ces constatations, nous avons voulu tenter, dans les deux derniers chapitres, de proposer une solution alternative plus valable.

Pour ce faire, nous avons, dans le chapitre 5, réalisé un certain nombre d'aménagements de la méthode SAMPO. Nous avons supprimé certains concepts, nous avons modifié la définition d'autres et nous avons élaboré une nouvelle modélisation de la méthode ainsi transformée, mais cette fois au moyen du modèle Entité/Association décrit dans [BOD89].

Ensuite, dans le chapitre 6, nous avons évoqué les principales caractéristiques que devrait posséder une nouvelle implantation, nous avons exploré quelques voies de résolution, et nous avons abordé le problème du schéma de la base de données si un langage traditionnel était utilisé. L'ampleur du travail requis par une implantation complète, efficace et conviviale ne nous a toutefois pas permis d'aller plus loin dans le cadre de ce mémoire.

Cela étant, nous croyons que si cette implantation était menée à terme de la façon dont nous l'avons envisagée, elle permettrait de résoudre les trois problèmes évoqués ci-avant

et constituerait un outil d'aide précieux pour celui qui voudrait spécifier un système d'information en utilisant la méthode SAMPO.

Quant à savoir si cette méthode va effectivement être utilisée, et bien qu'il s'agisse là d'une considération qui ne rentre pas directement dans le cadre de ce mémoire, nous restons sceptiques.

SAMPO possède, en effet, tous les concepts nécessaires pour être une bonne méthode d'analyse des communications ou du discours. Mais lorsque, partant du principe qu'un système d'information peut être considéré comme "un ensemble d'actes de langage pré-spécifiés (...)", cette méthode est appliquée pour spécifier des activités de bureau, certains de ces concepts deviennent inutiles, inappropriés ou définis de manière imprécise.

Nous avons tenté de remédier à ce problème dans le chapitre 5, mais nous restons néanmoins persuadés, comme [ROB89], qu'une spécification réalisée au moyen de SAMPO restera plus complexe et plus volumineuse que si une autre méthode avait été utilisée. Il suffit, pour s'en rendre compte, de regarder le volume de papier qui a été requis pour spécifier deux types de discours du cas Petitpas (cf. annexe A5).

Par ailleurs, une fois une spécification réalisée, il reste encore à l'interpréter et, si possible, à la transformer en un système informatique. Or ces deux processus font toujours, à l'heure actuelle, partie du domaine de la recherche car, comme le souligne [AUR88], "the transformation from a SAMPO specification to a technical design is complex and requires much work".

En conclusion, nous suggérerons donc, tout comme [ROB89], d'utiliser SAMPO plutôt comme un complément à d'autres méthodes plus classiques de spécification de systèmes d'information (pour voir apparaître, par exemple, la notion d'engagement) que comme une méthode autonome.

Les aménagements que nous avons apportés à cette méthode ainsi que l'implantation dont nous avons fait l'ébauche devraient, par ailleurs, rendre cette utilisation plus aisée et plus efficace.

## BIBLIOGRAPHIE

- [AUR88] AURAMÄKI E., LEHTINEN E., LYYTINEN K., *A Speech-Act-Based Office Modeling Approach*, ACM Transactions on Office Information Systems 6, N°2, pp 126-153, 1988.
- [AUS62] AUSTIN J.L., *How to do Things with Words*, Oxford, The University Press, 1962.
- [BIE80] BIERWISCH M., *Semantic Structure and Illocutionary Force*, Searle et al., 1980.
- [BOD89] BODART F., PIGNEUR Y., *Conception assistée des applications informatiques*, Masson, 1989.
- [BRE89] BRENNAN P., EMRICH M., GROCHOW J., CHIKOFISKY E., GIBSON M., *Case Directions*, Volume One, Number One, 1989.
- [CER69] CERF W., *Critical Review on How to Do Things with Words*, in : Symposium on J.L. Austin, 1969.
- [CHE76] CHEN P.P., *The Entity-Relationship Model : Toward a Unified View of Data*, ACM TODS, Vol. 1, N°1, 1976.
- [CLA81] CLARINVAL A., *Comprendre, connaître et maîtriser le Cobol*, Presses Universitaires de Namur, Première édition, 1981.
- [CLU89] McCLURE C., *The CASE Experience*, in : CASE in Depth, BYTE, avril 1989.
- [DIE89] DIEHL S., UDELL J., ALLEN D., WSZOLA S., JOCH A., APIKI S., GREHAN R., *Making a Case for CASE*, BYTE, décembre 1989.
- [GIA88] GIAVITTO J.L., DEVARENNE A., ROSUEL G., HOLVOET Y., *Adage : Utilisation de la Généricité pour*

*Construire des Environnements Adaptables*, Centre de recherches de la C.G.E., 1988.

- [HAI88] HAINAUT J.L., *Aspects théoriques et pratiques des modèles Entité-Association*, Institut d'Informatique, Facultés Universitaires de Namur, 1988.
- [ISD85] ISDOS Inc., *System Encyclopedia Manager, Langage Definition Manager : User Manual*, (SEM/LDM Version 1.4), 1985.
- [KAT77] KATZ J., *Propositionnal Structure and Illocutionary Force*, New York, Crowell and Co., 1977.
- [LAR87] LAROUSSE, *Petit Larousse en Couleurs*, Les éditions Françaises, 1987.
- [LEH86] LEHTINEN E., LYYTINEN K., *Action based model of information system*, Information Systems 11, N°4, pp 299-317, 1986.
- [LYY84a] LYYTINEN K., LEHTINEN E., *On information modelling through illocutionary logic*, Report of the third Scandinavian Research Seminar on Information Modelling and Data Base Management, University of Tampere, Finland, pp 35-116, 1984.
- [LYY84b] LYYTINEN K., LEHTINEN E., *Discourse analysis as an information system specification method*, Report of the seventh Scandinavian Research Seminar on Systemeering, School of Economics, Helsinki, pp 146-200, 1984.
- [MAS89] MASINI G., NAPOLI A., COLNET D., LEONARD D., TOMBRE K., *Les Langages à Objets*, InterEditions, 1989.
- [MET88] META SYSTEMS Ltd, *Model definition manager (MDM 2.2) : Internal user's guide*, 1988.



- [MET89a] META SYSTEMS Ltd, *Summary of MDM Constructs*, 1989.
- [MET89b] META SYSTEMS Ltd, *Quickspec user's Guide*, Version 1.0, 1989.
- [MET89c] META SYSTEMS Ltd, *Quickspec Language Guide*, Version 1.0, 1989.
- [ORR89] ORR K., GANE C., YOURDON E., CHEN P., CONSTANTINE L., *Methodology : The Experts Speak*, in : CASE in Depth, BYTE, avril 1989.
- [POS80] POSNER R., *Semantics and Pragmatics of Sentence Connectives in Natural Language*, Searle et al., 1980.
- [PUT88] PUTTRE M., *Can CASE Deliver ? Expectation vs. Reality*, Information Week, Issue 177, July 11, 1988.
- [ROB89] ROBAT D., THERASSE T., *Etude et évaluation d'une méthode de modélisation des communications dans une organisation : la méthode SAMPO*, Facultés Universitaires de Namur, 1989.
- [SAU88] SAUGE P., JEULIN P., *Graphtalk et Le Génie Logiciel*, Rank Xerox France, 1988.
- [SEA69] SEARLE J.R., *Speech acts*, Cambridge University Press, 1969.
- [SEA80] SEARLE J.R., KIEFER F., BIERWISCH M., (eds.), *Speech Act Theory and Pragmatics*, Publishing Company, Dordrecht, D.Reidel, 1980.
- [STA72] STALNAKER R.C., *Pragmatics*, in : Davidson and Harman (eds.), *Semantic of Natural Language*, Dordrecht, D.Reidel, 1972.

- [UER85] U.E.R GESTION, *Cas d'analyse : Firma "Petitpas"*,  
Institut d'Informatique, Facultés Universitaires de  
Namur, 1985.
- [WEL83] WELKE R.J., *IS/DSS : DBMS support for information  
systems development, Data Base Management : Theory  
and application*, pp195-250, 1983.
- [WEL88a] WELKE R.J., *The CASE repository : More than another  
database application*, Meta Systems Ltd, 1988.
- [WEL88b] WELKE R.J., *METABASE : A platform for the Next  
Generation of Meta Systems Products*, CASE Studies  
Conference, Meta Systems Ltd, 1988.
- [WUN80] WUNDERLICH D., *Methodological Remarks on Speech Act  
Theory*, in : Searle et al.

Facultés Universitaires Notre-Dame de la Paix à Namur

Institut d'Informatique

Année académique 1989-1990

CONCEPTION ET MISE EN OEUVRE DE  
MODELISATIONS ET D'IMPLANTATIONS  
DE LA METHODE SAMPO

ANNEXES

Mémoire de fin d'études présenté par

Serge CRAVATTE

Daniel WILLE

pour l'obtention du grade de licencié  
et maître en informatique

Promoteur : R. LESUISSE

ANNEXE A1

LISTES DE CRITERES DE SELECTION  
D'UN OUTIL CASE

# A CASE Tool Evaluation Checklist

*Use this checklist as a guide through the maze of CASE tool options*

## General information

- ☐ Vendor name
- ☐ Product name
- ☐ Date introduced
- ☐ Number of copies sold
- ☐ Price

## Hardware platform

- ☐ IBM PC and compatibles
- ☐ Macintosh
- ☐ Digital Equipment Corp.
- ☐ Unisys
- ☐ Honeywell
- ☐ Wang
- ☐ Apollo
- ☐ Sun
- ☐ Hewlett-Packard
- ☐ Other

## Category

- ☐ CASE toolkit
- ☐ CASE workbench

## Type of toolkit

- ☐ Planning
- ☐ Analysis
- ☐ Design
- ☐ Database design
- ☐ Real-time design
- ☐ Code generator
- ☐ Programming
- ☐ Maintenance
- ☐ Framework
- ☐ Project management

## Graphics

- ☐ Color
- ☐ Mouse
- ☐ Windows

## Error checking

- ☐ Syntax
- ☐ Consistency
- ☐ Completeness
- ☐ Requirements traceability
- ☐ Quality assurance

## Diagramming support

- ☐ Data flow
- ☐ Control flow
- ☐ Decision table/matrix
- ☐ Hierarchical tree structure
- ☐ Structure chart
- ☐ Action
- ☐ Warnier/Orr
- ☐ State transition
- ☐ Pseudocode
- ☐ Screen layout
- ☐ Dialogue flow
- ☐ Report layout
- ☐ Data structure
- ☐ Entity relationship
- ☐ Logical records
- ☐ Booch
- ☐ Petri nets
- ☐ Other

## Methodology support

- ☐ Yourdon
- ☐ DeMarco
- ☐ Gane/Sarson
- ☐ Bachman
- ☐ Chen
- ☐ Martin
- ☐ Merise
- ☐ Orr
- ☐ Jackson
- ☐ Ward/Mellor
- ☐ Hatley
- ☐ Object-oriented
- ☐ SADT
- ☐ Stradis
- ☐ Method-1
- ☐ LSDM
- ☐ Other

## Other requirements

- ☐ \_\_\_\_\_
- ☐ \_\_\_\_\_
- ☐ \_\_\_\_\_

## CASE repository

- ☐ Host-based
- ☐ PC-based
- ☐ DBMS architecture
- ☐ Reports
- ☐ Change control
- ☐ Audit trail
- ☐ Version control
- ☐ Download
- ☐ Logical partitioning
- ☐ Consolidation

## Prototyping support

- ☐ Screen painter
- ☐ Report painter
- ☐ Functional model
- ☐ Simulation

## Code generation

- ☐ Skeleton program
- ☐ Complete program
- ☐ Language
- ☐ On-line program
- ☐ Batch program

## Reengineering support

- ☐ Static analyzer
- ☐ Redocumentation
- ☐ Restructure
- ☐ Reverse engineering
- ☐ Dynamic analyzer
- ☐ Converter

## Life-cycle support

- ☐ Planning
- ☐ Analysis
- ☐ Design
- ☐ Implementation
- ☐ Maintenance
- ☐ Project management

## Target system

- ☐ On-line
- ☐ Batch
- ☐ Transaction processing
- ☐ Real-time
- ☐ Embedded

# A Guide to Selecting CASE Tools

## Editor's Note:

*Knowing the right questions to ask about CASE is as important as getting the right answers. In an article appearing in DATAMATION magazine, professor Michael L. Gibson (Auburn University, Auburn, Alabama) asked and answered some important questions that should help you in selecting the most appropriate CASE tool. Here are excerpts from the article.*

**Q:** *Is the tool a DBMS or a dictionary software system?*

**A:** Dictionary and database management systems provide greater integration capabilities. As a result, CASE tools with these underlying structures have a greater capacity for sharing specifications across functions.

**Q:** *What is the future direction and functionality of the tool?*

**A:** When evaluating CASE tools, remember that CASE systems development is still in its infancy, so don't reject a tool with valuable attributes just because it currently does not have the full capabilities that you want.

**Q:** *Does the tools manufacturer have an open architecture philosophy?*

**A:** A manufacturer's willingness to share file formats with all viable, noncompeting CASE manufacturers means that you can move smoothly from planning through to systems development because you will be able to integrate specifications across CASE components. Moreover, you will have a healthy variety of options for CASE software configurations. CASE manufacturers entering into exclusive hierarchical integration agreements with other non-competing CASE tool manufacturers ultimately limits choice.

**Q:** *Does the tool have an effective interface to other CASE design tools already purchased or under evaluation?*

**A:** Often, several methodologies are used to design a system, so it is important that a CASE tool provide a health array of methodological techniques to use in the process. The dictionary entries must be capable of being shared across these methodologies, so the dictionary should be strong and versatile.

**Q:** *Does the tool have graphical methodologies capable of "exploding" design diagrams and dictionary specifications to a reasonable depth?*

**A:** Most of the CASE design tools provide graphical methodologies for representing proposed systems design. The graphical

diagrams and the dictionary entries behind the components of the graphical diagrams must be capable of exploding to a reasonable number of lower, more specific levels.

**Q:** *Does the planning model in the CASE planning component provide comprehensive coverage of corporate and functional unit strategic planning and systems planning?*

**A:** The planning component contains a model for representing the corporation and for use in determining the direction of the corporation and systems development. The strength of the CASE top level components lies in the comprehensiveness of this planning model.

**Q:** *Does the CASE design tool provide analysis support for design documentation?*

**A:** This concerns the capacity of the CASE tool to analyze design documentation and determine if the specifications entered by the analyst conform to prescribed methodological rules. The analysis should also indicate where design dictionary entries are incomplete. For example, a DFD diagram with a freestanding block should be highlighted as violating one of the rules of structured methodology. In addition, blocks on a DFD not having a corresponding dictionary entry should be highlighted.

**Q:** *Does the tool have the capacity to generate design specification reports automatically?*

**A:** The specifications created during logical and physical design activities serve as a source of documentation for the system. While they are permanently stored on disk devices, it is often advisable to get hardcopy printouts of the design specifications for reference. Many CASE tools provide various report formats for this purpose, including the capability of indicating design flaws.

**Q:** *Does the tool permit distribution of design/development responsibilities?*

**A:** CASE design and development tools must provide a serviceable means of segregating job responsibilities and interfacing the individual efforts into a single system project.

**Q:** *Do the CASE design and development tools have the capacity to export portions of the design and development dictionary specifications?*

**A:** This is important, as design and development specifications for one system may be reusable in the design and development of other systems. "Reusable design" will join "reusable code" as a result of this capability.

**Q:** *Can the tool interface design and development specifications to the functional DBMS be used to maintain the company's data?*

**A:** It is rare to develop systems that are not affected by the database environment, and

the development of systems using CASE tools is no exception. Therefore, it is important that the CASE tools can interface design and development specifications of application systems to mainframe DBMSs and database creation or modification.

**Q:** *Does the tool enhance project management?*

**A:** The use of CASE tools does not preclude the need for effective project management. In fact, their use can enhance such management. Specifications that are entered using the planning component provide a boundary for design and development activities. This boundary provides a built-in means of determining when design and development activities diverge from originally planned specifications. Some CASE design and development tools can generate reports on the progress of individual project assignments and some can interface to existing project management software systems. Currently, this interface is a temporary exit from the CASE tool into the project management system, but the interface will become much stronger in the future and provide more automatic updating of the project schedule.

**Q:** *Is it possible to modify the CASE design and development tools relative to your firm's internal or existing methodology?*

**A:** CASE tools are prepackaged systems and may need modifications to make them

more suitable for individual installations. Thus, it is important that the system has the ability to add or delete menu options or to modify the system of graphical or dictionary entry screens.

**Q:** *Can the tool generate programs that span a range of systems?*

**A:** The hardware and software to create a transparent micro, mini, and mainframe environment are not far off. Consequently, the programs that the CASE tool generates must be able to provide the same execution services on a desktop micro as on a mighty mainframe. Some of today's CASE development systems already offer this.

It goes without saying that the CASE tool manufacturer should be willing to provide a list of installations using its software and grant permission to contact them. This major criterion should govern the purchase of any software system. Should a software vendor refuse to supply this information, you have reason to doubt the validity and comprehensiveness of its product.

Making intelligent CASE choices need not be mysterious. Considering the dividends that CASE can pay, the more informed your investment choices are, the greater your IS potential.

© 1988 Cahners Publishing Company. Excerpted from the July 1, 1988 issue of *DATA-MATION* with permission from the publisher.

ANNEXE A2

DICTIONNAIRE DES TYPES D'OBJETS



## DICTIONNAIRE DES TYPES D'OBJETS.

**ACTE** : Acte de langage (ACTEL) ou acte instrumental (ACTEI).

Propriétés :

- \* nom : le nom de l'acte ; obligatoire, string, non répétitif, identifiant.
- \* durée de vie : la durée de vie de l'acte (heures, jours ou mois) ; non obligatoire, string, non répétitif, non identifiant.
- \* nature : la nature de l'acte (interne ou externe) ; obligatoire, string, non répétitif, non identifiant.
- \* temps/fréquence : le moment (ou la fréquence) auquel (à laquelle) l'acte est exécuté ; non obligatoire, texte, non répétitif, identifiant.

**ACTE DE LANGAGE (ACTEL)** : Expression linguistique ayant une signification reconnue et non ambiguë. L'acte de langage est l'unité de base d'une communication. Il implique toujours au moins deux agents (le locuteur et l'auditeur qui, dans une situation extrême, peuvent être la même personne) et contient quatre "sous-actes" : l'acte d'expression, l'acte propositionnel, l'acte illocutoire et l'acte perlocutoire.

Propriétés :

- \* nom : le nom de l'acte de langage ; obligatoire, string, non répétitif, identifiant.
- \* place : la place de l'acte de langage dans la structure d'actes de langage ; obligatoire, string, non répétitif, non identifiant.
- \* type : le type de l'acte de langage (virtuel ou non virtuel) ; non obligatoire, string, non répétitif, non identifiant.
- \* monde possible : inclus toutes les caractéristiques des locuteurs, des auditeurs, du temps et des autres aspects de l'univers du discours ; non obligatoire, texte, non répétitif, non identifiant.

**ACTE D'EXPRESSION** : Acte exécuté par un locuteur lorsqu'il produit une expression. L'acte d'expression a trait à la forme donnée à l'expression.

Propriété :

\* forme : la forme de l'acte d'expression (graphique, vocale, gestuelle, ...) ; obligatoire, string, non répétitif, non identifiant.

**ACTE ILLOCUTOIRE** : Acte illocutoire élémentaire ou acte illocutoire complexe.

Propriété :

\* nom : le nom de l'acte illocutoire ; obligatoire, string, non répétitif, identifiant.

**ACTE ILLOCUTOIRE COMPLEXE** : Acte(s) illocutoire(s) élémentaire(s) ou complexe(s) relié(s) par un (des) connecteur(s) illocutoire(s) choisi(s) parmi la négation illocutoire, la conjonction illocutoire et la condition illocutoire.

Propriété :

\* contenu : le contenu de l'acte illocutoire complexe ; obligatoire, texte, non répétitif, non identifiant.

**ACTE ILLOCUTOIRE ELEMENTAIRE** : Unité de base d'une communication humaine significative, accomplie lorsqu'un locuteur formule des expressions avec une intention. Si l'intention est reconnue par l' (les) auditeur(s), un engagement est créé, qui lie les comportements futurs de toutes les parties. Un acte illocutoire élémentaire a deux composants : une force illocutoire et un contenu propositionnel.

Propriété :

\* intention : l'intention de l'acte illocutoire élémentaire ; obligatoire, texte, non répétitif, non identifiant.

**ACTE INSTRUMENTAL (ACTEI) :** Action humaine qui accomplit des changements dans le domaine entité. Son état initial et son résultat peuvent donc être décrits en faisant référence à des états du domaine entité.

Propriétés :

- \* nom : le nom de l'acte instrumental ; obligatoire, string, non répétitif, identifiant.
- \* but : le but de l'acte instrumental ; obligatoire, texte, non répétitif, non identifiant.
- \* arguments : les arguments de l'acte instrumental ; obligatoire, texte, non répétitif, non identifiant.

**ACTE PERLOCUTOIRE :** Acte accompli par un locuteur lorsqu'il dit quelque chose (acte illocutoire) et qu'un effet est produit sur les sentiments, les attitudes ou le comportement de l'(des) auditeur(s). Un acte perlocutoire engendre des effets qualifiés de perlocutoires.

Propriété :

- \* effets : les effets de l'acte perlocutoire sur les sentiments, les attitudes et le comportement de l' (des) auditeur(s) ; obligatoire, string, non répétitif, identifiant.

**ACTE PROPOSITIONNEL :** Acte par lequel le(s) locuteur(s) expriment le contenu propositionnel de l'acte illocutoire. Il s'agit d'un acte de dénotation et de prédiction.

Propriété :

- \* contenu : le contenu de l'acte propositionnel ; obligatoire, texte, non répétitif, non identifiant.

**ACTIVITE :** Routine d'action institutionnelle impliquant des actes de langage et des actes instrumentaux, et formée, à son tour, par des positions.

Propriétés :

- \* nom : le nom de l'activité ; obligatoire, string, non répétitif, identifiant.

- \* but : le but de l'activité ; obligatoire, texte, non répétitif, non identifiant.
- \* objectif : l'objectif de l'activité ; obligatoire, texte, non répétitif, non identifiant.
- \* problèmes : les problèmes ou ambiguïtés relatifs à l'activité ; obligatoire, texte, non répétitif, non identifiant.

**AGENT** : Personne, groupe de personnes ou machine qui accomplit un (des) acte(s).

Propriétés :

- \* nom : le nom de l'agent ; obligatoire, string, non répétitif, identifiant.
- \* type : le type de l'agent (individuel, groupe, humain ou machine) ; obligatoire, string, répétitif (3), non identifiant.

**ARRANGEMENTS ORGANISATIONNELS** : Contrats qui régissent l'implication d'un acte de langage ou d'un acte instrumental dans une activité.

Propriété :

- \* contenu : le contenu des arrangements organisationnels ; obligatoire, string, non répétitif, identifiant.

**CONVERSATION** : Partie d'un type de discours déclenchée par un prédicat de déclenchement.

Propriété :

- \* nom : le nom de la conversation ; obligatoire, string, non répétitif, identifiant.

**DISCOURS** : Tout effort soutenu de parole, c'est-à-dire toute séquence de phrases individuelles. En termes d'actes de langage, toute séquence d'actes de langage peut être considérée comme un discours.

**Propriété :**

- \* nom : le nom du discours ; obligatoire, string, non répétitif, identifiant.

**DOMAINE ACTION** : Comprend des entités dynamiques (actes) qui, d'ordinaire, ont une existence limitée dans le temps (bien que dans certains cas elles peuvent s'étendre sur de longues périodes) et qui peuvent introduire des changements dans le domaine entité.

**Propriété :**

- \* nom : le nom du domaine action ; obligatoire, string, non répétitif, identifiant.

**DOMAINE ENTITE** : Comprend des entités statiques qui persistent durant de longues périodes de temps.

**Propriété :**

- \* nom : le nom du domaine entité ; obligatoire, string, non répétitif, identifiant.

**ENGAGEMENT** : Conséquence d'un acte illocutoire élémentaire. Un engagement définit les attentes, en matière de comportements futurs des agents, qui vont lier les parties.

**Propriétés :**

- \* contenu : le contenu de l'engagement ; obligatoire, texte, non répétitif, non identifiant.
- \* type : le type de l'engagement (faible ou non faible) ; obligatoire, string, non répétitif, non identifiant.

**ENTITE (PASSIVE)** : Entité du discours contenue dans le domaine entité.

**Propriétés :**

- \* nom : le nom de l'entité ; obligatoire, string, non répétitif, identifiant.
- \* durée de vie : la durée de vie de l'entité (heures, jours ou mois) ; non obligatoire, string, non répétitif, non identifiant.

- \* propriétés : les propriétés de l'entité ; obligatoire, texte, non répétitif, non identifiant.

**FORCE ILLOCUTOIRE** : Composant principal de tout acte illocutoire. La force illocutoire détermine les relations sociales établies (engagements).

Propriétés :

- \* point illocutoire : un point ou un objectif interne et essentiel qui détermine le type de l'acte illocutoire associé. Un point illocutoire peut être assertif, commissif, directif, déclaratif ou expressif ; obligatoire, string, non répétitif, non identifiant.
- \* mode de réalisation du point illocutoire : la manière dont le point illocutoire peut être réalisé ; non obligatoire, texte, non répétitif, non identifiant.
- \* degré de force du point illocutoire : la force avec laquelle le point illocutoire est réalisé dans un acte de langage réussi ; non obligatoire, texte, non répétitif, non identifiant.
- \* conditions du contenu propositionnel : les conditions que le contenu d'un acte illocutoire doit satisfaire ; non obligatoire, texte, non répétitif, non identifiant.
- \* conditions préparatoires : les conditions qu'un locuteur doit supposer vérifiées (dans le monde) si un acte illocutoire donné doit être exécuté ; non obligatoire, texte, non répétitif, non identifiant.
- \* conditions de sincérité : les états psychologiques relatifs au contenu propositionnel ; non obligatoire, texte, non répétitif, non identifiant.
- \* degré de force des conditions de sincérité : la force avec laquelle les états psychologiques sont exprimés ; non obligatoire, texte, non répétitif, non identifiant.

**POSITION** : Définit qui est responsable des différentes phases d'une transaction.

Propriétés :

- \* nom : le nom de la position ; obligatoire, string, non répétitif, identifiant.

\* tâche : la tâche associée à la position ; non obligatoire, string, non répétitif, non identifiant.

**PREDICAT** : Prédicat de déclenchement, prédicat de porte ou prédicat de sélection.

Propriétés :

\* nom : le nom du prédicat ; obligatoire, string, non répétitif, identifiant.

\* numéro : le numéro du prédicat (T1, T2, S1, S2, ...) ; obligatoire, string, non répétitif, identifiant.

\* contenu : le contenu du prédicat ; obligatoire, texte, non répétitif, non identifiant.

**PREDICAT DE DECLENCHEMENT** : Détermine quand une conversation commence.

Propriété :

\* nom : le nom du prédicat de déclenchement ; obligatoire, string, non répétitif, identifiant.

**PREDICAT DE PORTE** : Permet l'exécution d'actes uniquement lorsque certaines conditions sont satisfaites.

Propriété :

\* nom : le nom du prédicat de porte ; obligatoire, string, non répétitif, identifiant.

**PREDICAT DE SELECTION** : Guide la sélection des actes dans une conversation.

Propriété :

\* nom : le nom du prédicat de sélection ; obligatoire, string, non répétitif, identifiant.

**SEGMENT DE DISCOURS** : Un groupe de "stages" dans une structure d'actes de langage. Les segments partagent une topicalisation commune et poursuivent un "sous-but" utile à la réalisation de l'objectif du type de discours associé.

**Propriété :**

- \* sous-but : le sous-but du segment de discours ; obligatoire, texte, non répétitif, non identifiant.

**STAGE** : Ensemble spécifique d'alternatives par lesquelles un acte de langage peut être suivi, et qui dépend de l'objectif du type de discours associé.

**Propriété :**

- \* objet : l'objectif du stage ; obligatoire, texte, non répétitif, non identifiant.

**STRUCTURE D'ACTES DE LANGAGE (SACTEL)** : Séquence ordonnée d'actes de langage qui forme un tout logique.

**Propriété :**

- \* nom : le nom de la structure d'actes de langage ; obligatoire, string, non répétitif, identifiant.

**TYPE DE DISCOURS** : La plus grande unité de communication qui peut être réalisée dans un système d'information de bureau et qui remplit un but social spécifique pouvant être identifié.

**Propriétés :**

- \* nom : le nom du type de discours ; obligatoire, string, non répétitif, identifiant.
- \* buts sociaux : les buts sociaux du type de discours ; obligatoire, texte, non répétitif, non identifiant.
- \* effets : les effets du type de discours ; obligatoire, texte, non répétitif, non identifiant.
- \* objectifs : les objectifs du type de discours ; obligatoire, texte, non répétitif, non identifiant.
- \* ambiguïtés : les ambiguïtés du type de discours ; obligatoire, texte, non répétitif, non identifiant.
- \* niveau : le niveau du type de discours (de base et/ou secondaire) ; obligatoire, string, répétitif (2), non identifiant.



ANNEXE A3

LISTING DES INSTRUCTIONS MDM

## LISTING DES INSTRUCTIONS MDM.

```
/******
/* Ecrit le 15 décembre 1989. */
/******
/* OBJETS */
```

```
OBJECT DISCOURSE;
 CODE 1;
```

```
OBJECT DISCOURSE_TYPE;
 CODE 2;
```

```
OBJECT CONVERSATION;
 CODE 3;
```

```
OBJECT PREDICATE;
 CODE 4;
```

```
OBJECT TRIGGERING_PREDICATE;
 CODE 5;
```

```
OBJECT SELECTION_PREDICATE;
 CODE 6;
```

```
OBJECT GATE_PREDICATE;
 CODE 7;
```

```
OBJECT SPEECH_ACT_PATTERN;
 CODE 8;
```

```
OBJECT SPEECH_ACT;
 CODE 9;
```

```
OBJECT INSTRUMENTAL_ACT;
 CODE 10;
```

```
OBJECT ACT;
 CODE 11;
```

OBJECT ENTITY\_DOMAIN;  
CODE 12;

OBJECT ENTITY;  
CODE 13;

OBJECT ACTION\_DOMAIN;  
CODE 14;

OBJECT ORGANIZATIONAL\_ARRANGEMENT;  
CODE 15;

OBJECT ACTIVITY;  
CODE 16;

OBJECT POSITION;  
CODE 17;

OBJECT AGENT;  
CODE 18;

OBJECT TOPICALIZATION;  
CODE 19;

OBJECT DISCOURSE\_SEGMENT;  
CODE 20;

OBJECT STAGE;  
CODE 21;

OBJECT UTTERANCE\_ACT;  
CODE 22;

OBJECT PROPOSITIONAL\_ACT;  
CODE 23;

OBJECT ELEMENTARY\_ILLOCUTIONARY\_ACT;  
CODE 24;

OBJECT COMPLEX\_ILLOCUTIONARY\_ACT;  
CODE 25;

OBJECT ILLOCUTIONARY\_ACT;  
CODE 26;

OBJECT PERLOCUTIONARY\_ACT;  
CODE 27;

OBJECT ILLOCUTIONARY\_FORCE;  
CODE 28;

OBJECT COMMITMENT;  
CODE 29;

/\* RELATIONS \*/

RELATIONSHIP to\_divide;  
PARTS is\_divided\_in ONE-ONE 3001,  
divides ONE-MANY 3002;  
COMBINATION is\_divided\_in DISCOURSE  
WITH divides DISCOURSE\_TYPE;  
CODE 1001;

RELATIONSHIP to\_split\_in;  
PARTS is\_split\_in ONE-ONE 3003,  
splits\_in ONE-MANY 3004;  
COMBINATION is\_split\_in DISCOURSE\_TYPE  
WITH splits\_in CONVERSATION;  
CODE 1002;

RELATIONSHIP to\_enclose;  
PARTS encloses ONE-ONE 3005,  
is\_enclosed\_in ONE-MANY 3006;  
COMBINATION encloses CONVERSATION  
WITH is\_enclosed\_in PREDICATE;  
CODE 1003;

RELATIONSHIP to\_be\_trig\_pred;  
PARTS is\_a\_trig\_pred ONE-ONE 3007,

|                                 |                       |
|---------------------------------|-----------------------|
| is_a_pred_1                     | ZERO-MANY 3008;       |
| COMBINATION is_a_trig_pred      | PREDICATE             |
| WITH is_a_pred_1                | TRIGGERING_PREDICATE; |
| CODE 1004;                      |                       |
|                                 |                       |
| RELATIONSHIP to_be_select_pred; |                       |
| PARTS is_a_select_pred          | ONE-ONE 3009,         |
| is_a_pred_2                     | ZERO-MANY 3010;       |
| COMBINATION is_a_select_pred    | PREDICATE             |
| WITH is_a_pred_2                | SELECTION_PREDICATE;  |
| CODE 1005;                      |                       |
|                                 |                       |
| RELATIONSHIP to_be_gate_pred;   |                       |
| PARTS is_a_gate_pred            | ONE-ONE 3011,         |
| is_a_pred_3                     | ZERO-MANY 3012;       |
| COMBINATION is_a_gate_pred      | PREDICATE             |
| WITH is_a_pred_3                | GATE_PREDICATE;       |
| CODE 1006;                      |                       |
|                                 |                       |
| RELATIONSHIP to_trigger;        |                       |
| PARTS triggers                  | ONE-ONE 3013,         |
| is_triggered_by                 | ONE-MANY 3014;        |
| COMBINATION triggers            | TRIGGERING_PREDICATE  |
| WITH is_triggered_by            | CONVERSATION;         |
| CODE 1007;                      |                       |
|                                 |                       |
| RELATIONSHIP to_describe_sactp; |                       |
| PARTS describes_sactp           | ONE-ONE 3015,         |
| is_described_by_1               | ZERO-MANY 3016;       |
| COMBINATION describes_sactp     | CONVERSATION          |
| WITH is_described_by_1          | SPEECH_ACT_PATTERN;   |
| CODE 1008;                      |                       |
|                                 |                       |
| RELATIONSHIP to_describe_iact;  |                       |
| PARTS describes_iact            | ONE-ONE 3017,         |
| is_described_by_2               | ZERO-MANY 3018;       |
| COMBINATION describes_iact      | CONVERSATION          |
| WITH is_described_by_2          | INSTRUMENTAL_ACT;     |
| CODE 1009;                      |                       |

|                                |                     |
|--------------------------------|---------------------|
| RELATIONSHIP to_order;         |                     |
| PARTS orders                   | ONE-ONE 3019,       |
| is_ordered_in                  | ONE-MANY 3020;      |
| COMBINATION orders             | SPEECH_ACT_PATTERN  |
| WITH is_ordered_in             | SPEECH_ACT;         |
| CODE 1010;                     |                     |
| RELATIONSHIP to_be_speech_act; |                     |
| PARTS is_a_speech_act          | ONE-ONE 3021,       |
| is_an_act_1                    | ZERO-MANY 3022;     |
| COMBINATION is_a_speech_act    | ACT                 |
| WITH is_an_act_1               | SPEECH_ACT;         |
| CODE 1011;                     |                     |
| RELATIONSHIP to_be_inst_act;   |                     |
| PARTS is_an_inst_act           | ONE-ONE 3023,       |
| is_an_act_2                    | ZERO-MANY 3024;     |
| COMBINATION is_an_inst_act     | ACT                 |
| WITH is_an_act_2               | INSTRUMENTAL_ACT;   |
| CODE 1012;                     |                     |
| RELATIONSHIP to_select;        |                     |
| PARTS selects                  | ZERO-MANY 3025,     |
| is_selected_by                 | ONE-MANY 3026;      |
| COMBINATION selects            | SELECTION_PREDICATE |
| WITH is_selected_by            | ACT;                |
| CODE 1013;                     |                     |
| RELATIONSHIP to_allow;         |                     |
| PARTS allows                   | ZERO-MANY 3027,     |
| is_allowed_by                  | ONE-MANY 3028;      |
| COMBINATION allows             | GATE_PREDICATE      |
| WITH is_allowed_by             | ACT;                |
| CODE 1014;                     |                     |
| RELATIONSHIP to_succeed;       |                     |
| PARTS succeeds                 | ZERO-MANY 3029,     |
| preceeds                       | ZERO-MANY 3030;     |
| COMBINATION succeeds           | ACT                 |
| WITH preceeds                  | ACT;                |

CODE 1015;

RELATIONSHIP to\_be\_simultaneous\_with;

|                             |                 |
|-----------------------------|-----------------|
| PARTS is_simul_with_1       | ZERO-MANY 3031, |
| is_simul_with_2             | ZERO-MANY 3032; |
| COMBINATION is_simul_with_1 | ACT             |
| WITH is_simul_with_2        | ACT;            |

CODE 1016;

RELATIONSHIP to\_change;

|                     |                |
|---------------------|----------------|
| PARTS changes       | ONE-MANY 3033, |
| is_changed_by       | ZERO-ONE 3034; |
| COMBINATION changes | ACT            |
| WITH is_changed_by  | ENTITY_DOMAIN; |

CODE 1017;

RELATIONSHIP to\_comprise;

|                       |                |
|-----------------------|----------------|
| PARTS comprises       | ONE-ONE 3035,  |
| is_comprised_in       | ONE-MANY 3036; |
| COMBINATION comprises | ENTITY_DOMAIN  |
| WITH is_comprised_in  | ENTITY;        |

CODE 1018;

RELATIONSHIP to\_include;

|                      |                |
|----------------------|----------------|
| PARTS includes       | ONE-ONE 3037,  |
| is_included_in       | ONE-MANY 3038; |
| COMBINATION includes | ACTION_DOMAIN  |
| WITH is_included_in  | ACT;           |

CODE 1019;

RELATIONSHIP to\_involve;

|                            |                            |
|----------------------------|----------------------------|
| PARTS is_involved_in       | ONE-MANY 3039,             |
| governs                    | ONE-MANY 3040,             |
| involves                   | ONE-MANY 3041;             |
| COMBINATION is_involved_in | ACT                        |
| WITH governs               | ORGANIZATIONAL_ARRANGEMENT |
| WITH involves              | ACTIVITY;                  |

CODE 1020;

## RELATIONSHIP to\_form;

PARTS is\_formed\_by  
forms

COMBINATION is\_formed\_by  
WITH forms

CODE 1021;

ONE-MANY 3042,

ONE-MANY 3043;

ACTIVITY

POSITION;

## RELATIONSHIP to\_hold;

PARTS is\_held\_by  
holds

COMBINATION is\_held\_by  
WITH holds

CODE 1022;

ONE-MANY 3044,

ZERO-MANY 3045;

POSITION

AGENT;

## RELATIONSHIP to\_speak;

PARTS speaks  
is\_spoken\_by

COMBINATION speaks  
WITH is\_spoken\_by

CODE 1023;

ONE-MANY 3046,

ZERO-MANY 3047;

AGENT

SPEECH\_ACT;

## RELATIONSHIP to\_hear;

PARTS hears  
is\_heard\_by

COMBINATION hears  
WITH is\_heard\_by

CODE 1024;

ONE-MANY 3048,

ZERO-MANY 3049;

AGENT

SPEECH\_ACT;

## RELATIONSHIP to\_perform;

PARTS performs  
is\_performed\_by

COMBINATION performs  
WITH is\_performed\_by

CODE 1025;

ONE-MANY 3050,

ONE-MANY 3051;

AGENT

ACT;

## RELATIONSHIP to\_concern;

PARTS concerns  
is\_concerned\_by

COMBINATION concerns  
WITH is\_concerned\_by

ONE-ONE 3052,

ONE-MANY 3053;

TOPICALIZATION

SPEECH\_ACT;



CODE 1026;

RELATIONSHIP to\_characterize;

|                           |                    |
|---------------------------|--------------------|
| PARTS characterizes       | ONE-ONE 3054,      |
| is_characterized_by       | ONE-MANY 3055;     |
| COMBINATION characterizes | TOPICALIZATION     |
| WITH is_characterized_by  | DISCOURSE_SEGMENT; |

CODE 1027;

RELATIONSHIP to\_group;

|                    |                   |
|--------------------|-------------------|
| PARTS groups       | ONE-ONE 3056,     |
| is_grouped_in      | ONE-MANY 3057;    |
| COMBINATION groups | DISCOURSE_SEGMENT |
| WITH is_grouped_in | STAGE;            |

CODE 1028;

RELATIONSHIP to\_activate;

|                             |                |
|-----------------------------|----------------|
| PARTS is_activated_by       | ZERO-ONE 3058, |
| activates                   | ONE-MANY 3059; |
| COMBINATION is_activated_by | STAGE          |
| WITH activates              | SPEECH_ACT;    |

CODE 1029;

RELATIONSHIP to\_contain;

|                      |                    |
|----------------------|--------------------|
| PARTS contains       | ONE-ONE 3060,      |
| is_contained_in      | ONE-MANY 3061;     |
| COMBINATION contains | SPEECH_ACT_PATTERN |
| WITH is_contained_in | STAGE;             |

CODE 1030;

RELATIONSHIP to\_suggest;

|                      |                |
|----------------------|----------------|
| PARTS suggests       | ONE-ONE 3062,  |
| is_suggested_by      | ONE-MANY 3063; |
| COMBINATION suggests | DISCOURSE_TYPE |
| WITH is_suggested_by | STAGE;         |

CODE 1031;

RELATIONSHIP to\_be\_utterance\_act;

|                     |                |
|---------------------|----------------|
| PARTS is_an_utt_act | ONE-ONE 3064,  |
| is_a_sact_1         | ONE-MANY 3065; |

COMBINATION is\_an\_utt\_act                    SPEECH\_ACT  
              WITH is\_a\_sact\_1                UTTERANCE\_ACT;  
CODE 1032;

RELATIONSHIP to\_be\_propositional\_act;  
PARTS is\_a\_prop\_act                    ONE-ONE    3066,  
              is\_a\_sact\_2                    ZERO-MANY 3067;  
COMBINATION is\_a\_prop\_act                SPEECH\_ACT  
              WITH is\_a\_sact\_2                PROPOSITIONAL\_ACT;  
CODE 1033;

RELATIONSHIP to\_be\_illocutionary\_act;  
PARTS is\_an\_ill\_act                    ONE-ONE    3068,  
              is\_a\_sact\_3                    ZERO-MANY 3069;  
COMBINATION is\_an\_ill\_act                SPEECH\_ACT  
              WITH is\_a\_sact\_3                ILLOCUTIONARY\_ACT;  
CODE 1034;

RELATIONSHIP to\_be\_perlocutionary\_act;  
PARTS is\_a\_perl\_act                    ONE-ONE    3070,  
              is\_a\_sact\_4                    ZERO-MANY 3071;  
COMBINATION is\_a\_perl\_act                SPEECH\_ACT  
              WITH is\_a\_sact\_4                PERLOCUTIONARY\_ACT;  
CODE 1035;

RELATIONSHIP to\_be\_elementary\_ill\_act;  
PARTS is\_an\_elem\_ill\_act                ONE-ONE    3072,  
              is\_an\_ill\_act\_1                ZERO-MANY 3073;  
COMBINATION is\_an\_elem\_ill\_act            ILLOCUTIONARY\_ACT  
              WITH is\_an\_ill\_act\_1  
ELEMENTARY\_ILLOCUTIONARY\_ACT;  
CODE 1036;

RELATIONSHIP to\_be\_complex\_ill\_act;  
PARTS is\_a\_compl\_ill\_act                ONE-ONE    3074,  
              is\_an\_ill\_act\_2                ZERO-MANY 3075;  
COMBINATION is\_a\_compl\_ill\_act            ILLOCUTIONARY\_ACT  
              WITH is\_an\_ill\_act\_2            COMPLEX\_ILLOCUTIONARY\_ACT;  
CODE 1037;

## RELATIONSHIP to\_determine;

|                        |                     |
|------------------------|---------------------|
| PARTS determines       | ONE-ONE 3076,       |
| implies                | ONE-MANY 3077,      |
| is_determined_by       | ONE-MANY 3078;      |
| COMBINATION determines | ILLOCUTIONARY_FORCE |
| WITH implies           |                     |

## ELEMENTARY\_ILLOCUTIONARY\_ACT

|                       |             |
|-----------------------|-------------|
| WITH is_determined_by | COMMITMENT; |
|-----------------------|-------------|

CODE 1038;

## RELATIONSHIP to\_represent\_elem\_ill\_act;

|                              |                    |
|------------------------------|--------------------|
| PARTS rep_elem_ill_act       | ZERO-ONE 3079,     |
| is_rep_by_per_act            | ONE-MANY 3080;     |
| COMBINATION rep_elem_ill_act | PERLOCUTIONARY_ACT |
| WITH is_rep_by_per_act       |                    |

## ELEMENTARY\_ILLOCUTIONARY\_ACT;

CODE 1039;

## RELATIONSHIP to\_represent\_proposit\_act;

|                          |                |
|--------------------------|----------------|
| PARTS rep_prop_act       | ZERO-ONE 3081, |
| is_rep_by_el_ill_act     | ONE-MANY 3082; |
| COMBINATION rep_prop_act |                |

## ELEMENTARY\_ILLOCUTIONARY\_ACT

|                           |                    |
|---------------------------|--------------------|
| WITH is_rep_by_el_ill_act | PROPOSITIONAL_ACT; |
|---------------------------|--------------------|

CODE 1040;

## RELATIONSHIP to\_represent\_utterance\_act;

|                         |                   |
|-------------------------|-------------------|
| PARTS rep_utt_act       | ZERO-ONE 3083,    |
| is_rep_by_prop_act      | ONE-MANY 3084;    |
| COMBINATION rep_utt_act | PROPOSITIONAL_ACT |
| WITH is_rep_by_prop_act | UTTERANCE_ACT;    |

CODE 1041;

## RELATIONSHIP to\_compose\_c\_and\_c;

|                          |                            |
|--------------------------|----------------------------|
| PARTS comp_c_and_c       | ZERO-MANY 3085,            |
| is_comp_by_c_and_c       | ZERO-ONE 3086;             |
| COMBINATION comp_c_and_c | COMPLEX_ILLOCUTIONARY_ACT  |
| WITH is_comp_by_c_and_c  | COMPLEX_ILLOCUTIONARY_ACT; |

CODE 1042;

```

RELATIONSHIP to_compose_e_and_c;
 PARTS comp1_e_and_c ZERO-MANY 3087,
 comp2_e_and_c ZERO-MANY 3088,
 is_comp_by_e_and_c ZERO-ONE 3089;
 COMBINATION comp1_e_and_c
ELEMENTARY_ILLOCUTIONARY_ACT
 WITH comp2_e_and_c COMPLEX_ILLOCUTIONARY_ACT
 WITH is_comp_by_e_and_c COMPLEX_ILLOCUTIONARY_ACT;
CODE 1043;

```

```

RELATIONSHIP to_compose_e_and_e;
 PARTS comp_e_and_e ZERO-MANY 3090,
 is_comp_by_e_and_e ZERO-ONE 3091;
 COMBINATION comp_e_and_e
ELEMENTARY_ILLOCUTIONARY_ACT
 WITH is_comp_by_e_and_e COMPLEX_ILLOCUTIONARY_ACT;
CODE 1044;

```

```

RELATIONSHIP to_form_not_c;
 PARTS forms_not_c ZERO-MANY 3092,
 is_formed_by_not_c ZERO-ONE 3093;
 COMBINATION forms_not_c COMPLEX_ILLOCUTIONARY_ACT
 WITH is_formed_by_not_c COMPLEX_ILLOCUTIONARY_ACT;
CODE 1045;

```

```

RELATIONSHIP to_form_not_e;
 PARTS forms_not_e ZERO-MANY 3094,
 is_formed_by_not_e ZERO-ONE 3095;
 COMBINATION forms_not_e
ELEMENTARY_ILLOCUTIONARY_ACT
 WITH is_formed_by_not_e COMPLEX_ILLOCUTIONARY_ACT;
CODE 1046;

```

```

RELATIONSHIP to_implicate_e_p;
 PARTS implicates_e_p ZERO-MANY 3096,
 is_implicated_by_e_p ZERO-ONE 3097,
 is_formed_by_e_p ZERO-ONE 3098;
 COMBINATION implicates_e_p PREDICATE
 WITH is_implicated_by_e_p
ELEMENTARY_ILLOCUTIONARY_ACT

```

WITH is\_formed\_by\_e\_p COMPLEX\_ILLOCUTIONARY\_ACT;  
CODE 1047;

RELATIONSHIP to\_implicate\_c\_p;

PARTS implicates\_c\_p ZERO-MANY 3099,  
is\_implicated\_by\_c\_p ZERO-ONE 3100,  
is\_formed\_by\_c\_p ZERO-ONE 3101;  
COMBINATION implicates\_c\_p PREDICATE  
WITH is\_implicated\_by\_c\_p COMPLEX\_ILLOCUTIONARY\_ACT  
WITH is\_formed\_by\_c\_p COMPLEX\_ILLOCUTIONARY\_ACT;  
CODE 1048;

/\* PROPRIETES \*/

PROPERTY GOALS;

APPLIES-OBJECT DISCOURSE\_TYPE,  
INSTRUMENTAL\_ACT,  
ACTIVITY;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2001;  
CASE-SENSITIVE;

PROPERTY EFFECTS;

APPLIES-OBJECT DISCOURSE\_TYPE,  
PERLOCUTIONARY\_ACT;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2002;  
CASE-SENSITIVE;

PROPERTY OBJECTIVES;

APPLIES-OBJECT DISCOURSE\_TYPE;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2003;  
CASE-SENSITIVE;

PROPERTY AMBIGUITIES;

APPLIES-OBJECT DISCOURSE\_TYPE;

NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2004;  
CASE-SENSITIVE;

PROPERTY LEVEL;  
APPLIES-OBJECT DISCOURSE\_TYPE;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2005;  
CASE-SENSITIVE;

PROPERTY NUMBER;  
APPLIES-OBJECT PREDICATE;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2006;  
CASE-SENSITIVE;

PROPERTY CONTENT;  
APPLIES-OBJECT PREDICATE,  
PROPOSITIONAL\_ACT,  
COMPLEX\_ILLOCUTIONARY\_ACT,  
COMMITMENT,  
ORGANIZATIONAL\_ARRANGEMENT;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2007;  
CASE-SENSITIVE;

PROPERTY ARGUMENTS;  
APPLIES-OBJECT INSTRUMENTAL\_ACT;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2008;  
CASE-SENSITIVE;

PROPERTY PLACE;  
APPLIES-OBJECT SPEECH\_ACT;  
NUMBER-VALUES ZERO-ONE;

VALUES TEXT;  
CODE 2009;  
CASE-SENSITIVE;

PROPERTY TYPE;  
APPLIES-OBJECT SPEECH\_ACT,  
COMMITMENT,  
AGENT;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2010;  
CASE-SENSITIVE;

PROPERTY POSSIBLE\_WORLD;  
APPLIES-OBJECT SPEECH\_ACT;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2011;  
CASE-SENSITIVE;

PROPERTY LIFE\_PERIOD;  
APPLIES-OBJECT ACT,  
ENTITY;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2012;  
CASE-SENSITIVE;

PROPERTY NATURE;  
APPLIES-OBJECT ACT;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2013;  
CASE-SENSITIVE;

PROPERTY basic-name;  
APPLIES-OBJECT DISCOURSE,  
DISCOURSE\_TYPE,  
CONVERSATION,  
PREDICATE,

TRIGGERING\_PREDICATE,  
SELECTION\_PREDICATE,  
GATE\_PREDICATE,  
INSTRUMENTAL\_ACT,  
SPEECH\_ACT,  
ACT,  
ILLOCUTIONARY\_ACT,  
ENTITY\_DOMAIN,  
ENTITY,  
ACTION\_DOMAIN,  
ACTIVITY,  
POSITION,  
AGENT,  
ORGANIZATIONAL\_ARRANGEMENT,  
TOPICALIZATION,  
DISCOURSE\_SEGMENT,  
STAGE,  
UTTERANCE\_ACT,  
PROPOSITIONAL\_ACT,  
ELEMENTARY\_ILLOCUTIONARY\_ACT,  
COMPLEX\_ILLOCUTIONARY\_ACT,  
PERLOCUTIONARY\_ACT,  
ILLOCUTIONARY\_FORCE,  
COMMITMENT,  
SPEECH\_ACT\_PATTERN;  
  
NUMBER-VALUES ONE-ONE;  
VALUES STRING;  
CODE 2014;  
IDENTIFIER;  
CASE-SENSITIVE;  
  
PROPERTY TIME\_FREQUENCY;  
APPLIES-OBJECT ACT;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2020;  
CASE-SENSITIVE;  
  
PROPERTY FORM;  
APPLIES-OBJECT UTTERANCE\_ACT;



NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2021;  
CASE-SENSITIVE;

PROPERTY INTENTION;  
APPLIES-OBJECT ELEMENTARY\_ILLOCUTIONARY\_ACT;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2022;  
CASE-SENSITIVE;

PROPERTY ILLOCUTIONARY\_POINT;  
APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2023;  
CASE-SENSITIVE;

PROPERTY MODE\_OF\_ACHIEV\_OF\_ILL\_POINT;  
APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2024;  
CASE-SENSITIVE;

PROPERTY DEGREE\_OF\_STRENGTH\_ILL\_POINT;  
APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
NUMBER-VALUES ZERO-ONE;  
VALUES TEXT;  
CODE 2025;  
CASE-SENSITIVE;

PROPERTY PROPOSITIONAL\_CONDITIONS;  
APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
NUMBER-VALUES ZERO-MANY;  
VALUES TEXT;  
CODE 2026;  
CASE-SENSITIVE;

PROPERTY PREPARATORY\_CONDITIONS;  
 APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
 NUMBER-VALUES ZERO-MANY;  
 VALUES TEXT;  
 CODE 2027;  
 CASE-SENSITIVE;

PROPERTY SINCERITY\_CONDITIONS;  
 APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
 NUMBER-VALUES ZERO-MANY;  
 VALUES TEXT;  
 CODE 2028;  
 CASE-SENSITIVE;

PROPERTY DEGREE\_OF\_STRENGTH\_SINC\_COND;  
 APPLIES-OBJECT ILLOCUTIONARY\_FORCE;  
 NUMBER-VALUES ZERO-MANY;  
 VALUES TEXT;  
 CODE 2029;  
 CASE-SENSITIVE;

PROPERTY PROPERTIES;  
 APPLIES-OBJECT ENTITY;  
 NUMBER-VALUES ZERO-MANY;  
 VALUES TEXT;  
 CODE 2030;  
 CASE-SENSITIVE;

PROPERTY TARGET;  
 APPLIES-OBJECT ACTIVITY;  
 NUMBER-VALUES ZERO-MANY;  
 VALUES TEXT;  
 CODE 2031;  
 CASE-SENSITIVE;

PROPERTY PROBLEMS;  
 APPLIES-OBJECT ACTIVITY;  
 NUMBER-VALUES ZERO-MANY;  
 VALUES TEXT;  
 CODE 2032;

CASE-SENSITIVE;

PROPERTY TASK;

APPLIES-OBJECT POSITION;

NUMBER-VALUES ZERO-MANY;

VALUES TEXT;

CODE 2033;

CASE-SENSITIVE;

PROPERTY ILL\_FORCE;

APPLIES-OBJECT TOPICALIZATION;

NUMBER-VALUES ZERO-MANY;

VALUES TEXT;

CODE 2034;

CASE-SENSITIVE;

PROPERTY PROPOSITIONAL\_CONTENT;

APPLIES-OBJECT TOPICALIZATION;

NUMBER-VALUES ZERO-MANY;

VALUES TEXT;

CODE 2035;

CASE-SENSITIVE;

PROPERTY SUBGOAL;

APPLIES-OBJECT DISCOURSE\_SEGMENT;

NUMBER-VALUES ZERO-MANY;

VALUES TEXT;

CODE 2036;

CASE-SENSITIVE;

PROPERTY PURPOSE;

APPLIES-OBJECT STAGE;

NUMBER-VALUES ZERO-MANY;

VALUES TEXT;

CODE 2037;

CASE-SENSITIVE;

ANNEXE A4

LISTING DE LA TABLE XT

## LISTING DE LA TABLE XT.

```
/*
Ecrit le 15 décembre 1989.
*/
OBJETS
*/
```

```
OBJECT DISCOURSE;
TAG "Discourse";
```

```
OBJECT DISCOURSE_TYPE;
TAG "Discourse Type";
```

```
OBJECT CONVERSATION;
TAG "Conversation";
```

```
OBJECT PREDICATE;
TAG "Predicate";
```

```
OBJECT TRIGGERING_PREDICATE;
TAG "Triggering Predicate";
```

```
OBJECT SELECTION_PREDICATE;
TAG "Selection Predicate";
```

```
OBJECT GATE_PREDICATE;
TAG "Gate Predicate";
```

```
OBJECT SPEECH_ACT_PATTERN;
TAG "Speech Act Pattern";
```

```
OBJECT SPEECH_ACT;
TAG "Speech Act";
```

```
OBJECT INSTRUMENTAL_ACT;
TAG "Instrumental Act";
```

```
OBJECT ACT;
TAG "Act";
```

OBJECT ENTITY\_DOMAIN;  
TAG "Entity Domain";

OBJECT ENTITY;  
TAG "Entity";

OBJECT ACTION\_DOMAIN;  
TAG "Action Domain";

OBJECT ORGANIZATIONAL\_ARRANGEMENT;  
TAG "Organisational Arrangement";

OBJECT ACTIVITY;  
TAG "Activity";

OBJECT POSITION;  
TAG "Position";

OBJECT AGENT;  
TAG "Agent";

OBJECT TOPICALIZATION;  
TAG "Topicalization";

OBJECT DISCOURSE\_SEGMENT;  
TAG "Discourse Segment";

OBJECT STAGE;  
TAG "Stage";

OBJECT UTTERANCE\_ACT;  
TAG "Utterance Act";

OBJECT PROPOSITIONAL\_ACT;  
TAG "Propositional Act";

OBJECT ELEMENTARY\_ILLOCUTIONARY\_ACT;  
TAG "Elementary Illocutionary Act";

OBJECT COMPLEX\_ILLOCUTIONARY\_ACT;  
TAG "Complex Illocutionary Act";

OBJECT ILLOCUTIONARY\_ACT;  
TAG "Illocutionary Act";

OBJECT PERLOCUTIONARY\_ACT;  
TAG "Perlocutionary Act";

OBJECT ILLOCUTIONARY\_FORCE;  
TAG "Illocutionary Force";

OBJECT COMMITMENT;  
TAG "Commitment";

/\* RELATIONS \*/

RELATIONSHIP to\_divide;  
TAG FOR is\_divided\_in "Divided in : " WITH  
ROLE 1 divides "Divides" ;  
TAG FOR divides "Divides : " WITH  
ROLE 1 is\_divided\_in "Divided in" ;

RELATIONSHIP to\_split\_in;  
TAG FOR is\_split\_in "Split in : " WITH  
ROLE 1 splits\_in "Splits in" ;  
TAG FOR splits\_in "Splits in : " WITH  
ROLE 1 is\_split\_in "Split in" ;

RELATIONSHIP to\_enclose;  
TAG FOR encloses "Encloses : " WITH  
ROLE 1 is\_enclosed\_in "Enclosed in" ;  
TAG FOR is\_enclosed\_in "Enclosed in: " WITH  
ROLE 1 encloses "Encloses" ;

RELATIONSHIP to\_be\_trig\_pred;  
TAG FOR is\_a\_trig\_pred "Triggering Predicate : " WITH  
ROLE 1 is\_a\_pred\_1 "Predicate" ;  
TAG FOR is\_a\_pred\_1 "Predicate : " WITH  
ROLE 1 is\_a\_trig\_pred "Triggering Predicate" ;

RELATIONSHIP to\_be\_select\_pred;

TAG FOR is\_a\_select\_pred "Selection Predicate : " WITH

ROLE 1 is\_a\_pred\_2 "Predicate" ;

TAG FOR is\_a\_pred\_2 "Predicate. : " WITH

ROLE 1 is\_a\_select\_pred "Selection Predicate" ;

RELATIONSHIP to\_be\_gate\_pred;

TAG FOR is\_a\_gate\_pred "Gate Predicate : " WITH

ROLE 1 is\_a\_pred\_3 "Predicate" ;

TAG FOR is\_a\_pred\_3 "Predicate.. : " WITH

ROLE 1 is\_a\_gate\_pred "Gate Predicate" ;

RELATIONSHIP to\_trigger;

TAG FOR triggers "Triggers : " WITH

ROLE 1 is\_triggered\_by "Triggered by" ;

TAG FOR is\_triggered\_by "Triggered by : " WITH

ROLE 1 triggers "Triggers" ;

RELATIONSHIP to\_describe\_sactp;

TAG FOR describes\_sactp "Describes Speech Act Pat. : " WITH

ROLE 1 is\_described\_by\_1 "Described by" ;

TAG FOR is\_described\_by\_1 "Described by : " WITH

ROLE 1 describes\_sactp "Describes Speech Act Pat." ;

RELATIONSHIP to\_describe\_iact;

TAG FOR describes\_iact "Describes Instrumental Acts : " WITH

ROLE 1 is\_described\_by\_2 "Described by" ;

TAG FOR is\_described\_by\_2 "Described by. : " WITH

ROLE 1 describes\_iact "Describes Instrumental Acts" ;

RELATIONSHIP to\_order;

TAG FOR orders "Orders : " WITH

ROLE 1 is\_ordered\_in "Ordered in" ;

TAG FOR is\_ordered\_in "Ordered in : " WITH

ROLE 1 orders "Orders" ;

RELATIONSHIP to\_be\_speech\_act;

TAG FOR is\_a\_speech\_act "Speech Act : " WITH

ROLE 1 is\_an\_act\_1 "Act" ;

TAG FOR is\_an\_act\_1 "Act : " WITH



ROLE 1 is\_a\_speech\_act "Speech Act" ;

RELATIONSHIP to\_be\_inst\_act;

TAG FOR is\_an\_inst\_act "Instrumental Act : " WITH

ROLE 1 is\_an\_act\_2 "Act" ;

TAG FOR is\_an\_act\_2 "Act. : " WITH

ROLE 1 is\_an\_inst\_act "Instrumental Act" ;

RELATIONSHIP to\_select;

TAG FOR selects "Selects : " WITH

ROLE 1 is\_selected\_by "Selected by" ;

TAG FOR is\_selected\_by "Selected by : " WITH

ROLE 1 selects "Selects" ;

RELATIONSHIP to\_allow;

TAG FOR allows "Allows : " WITH

ROLE 1 is\_allowed\_by "Allowed by" ;

TAG FOR is\_allowed\_by "Allowed by : " WITH

ROLE 1 allows "Allows" ;

RELATIONSHIP to\_succeed;

TAG FOR succeeds "Succeeds : " WITH

ROLE 1 preceeds "Preceeds" ;

TAG FOR preceeds "Preceeds : " WITH

ROLE 1 succeeds "Succeeds" ;

RELATIONSHIP to\_be\_simultaneous\_with;

TAG FOR is\_simul\_with\_1 "Simultaneous with : " WITH

ROLE 1 is\_simul\_with\_2 "Simultaneous with" ;

RELATIONSHIP to\_change;

TAG FOR changes "Changes : " WITH

ROLE 1 is\_changed\_by "Changed by" ;

TAG FOR is\_changed\_by "Changed by : " WITH

ROLE 1 changes "Changes" ;

RELATIONSHIP to\_comprise;

TAG FOR comprises "Comprises : " WITH

ROLE 1 is\_comprised\_in "Comprised in" ;

TAG FOR is\_comprised\_in "Comprised in : " WITH

ROLE 1 comprises "Comprises" ;

RELATIONSHIP to\_include;

TAG FOR includes "Includes : " WITH

ROLE 1 is\_included\_in "Included in" ;

TAG FOR is\_included\_in "Included in : " WITH

ROLE 1 includes "Includes" ;

RELATIONSHIP to\_involve;

TAG FOR is\_involved\_in "Involved in : " WITH

ROLE 1 governs "Governs" ,

ROLE 2 involves "Involves" ;

TAG FOR governs "Governs : " WITH

ROLE 1 is\_involved\_in "Involved in" ,

ROLE 2 involves "Involves" ;

TAG FOR involves "Involves : " WITH

ROLE 1 is\_involved\_in "Involved in" ,

ROLE 2 governs "Governs" ;

RELATIONSHIP to\_form;

TAG FOR is\_formed\_by "Formed by : " WITH

ROLE 1 forms "Forms" ;

TAG FOR forms "Forms : " WITH

ROLE 1 is\_formed\_by "Formed by" ;

RELATIONSHIP to\_hold;

TAG FOR is\_held\_by "Held by : " WITH

ROLE 1 holds "Holds" ;

TAG FOR holds "Holds : " WITH

ROLE 1 is\_held\_by "Held by" ;

RELATIONSHIP to\_speak;

TAG FOR speaks "Speaks : " WITH

ROLE 1 is\_spoken\_by "Spoken by" ;

TAG FOR is\_spoken\_by "Spoken by : " WITH

ROLE 1 speaks "Speaks" ;

RELATIONSHIP to\_hear;

TAG FOR hears "Hears : " WITH

ROLE 1 is\_heard\_by "Heard by" ;

TAG FOR is\_heard\_by "Heard by : " WITH  
ROLE 1 hears "Hears" ;

RELATIONSHIP to\_perform;

TAG FOR performs "Performs : " WITH  
ROLE 1 is\_performed\_by "Performed by" ;  
TAG FOR is\_performed\_by "Performed by : " WITH  
ROLE 1 performs "Performs" ;

RELATIONSHIP to\_concern;

TAG FOR concerns "Concerns : " WITH  
ROLE 1 is\_concerned\_by "Concerned by" ;  
TAG FOR is\_concerned\_by "Concerned by : " WITH  
ROLE 1 concerns "Concerns" ;

RELATIONSHIP to\_characterize;

TAG FOR characterizes "Characterizes : " WITH  
ROLE 1 is\_characterized\_by "Characterized by" ;  
TAG FOR is\_characterized\_by "Characterized by : " WITH  
ROLE 1 characterizes "Characterizes" ;

RELATIONSHIP to\_group;

TAG FOR groups "Groups : " WITH  
ROLE 1 is\_grouped\_in "Grouped in" ;  
TAG FOR is\_grouped\_in "Grouped in : " WITH  
ROLE 1 groups "Groups" ;

RELATIONSHIP to\_activate;

TAG FOR is\_activated\_by "Activated by : " WITH  
ROLE 1 activates "Activates" ;  
TAG FOR activates "Activates : " WITH  
ROLE 1 is\_activated\_by "Activated by" ;

RELATIONSHIP to\_contain;

TAG FOR contains "Contains : " WITH  
ROLE 1 is\_contained\_in "Contained in" ;  
TAG FOR is\_contained\_in "Contained in : " WITH  
ROLE 1 contains "Contains" ;

RELATIONSHIP to\_suggest;

TAG FOR suggests "Suggests : " WITH  
ROLE 1 is\_suggested\_by "Suggested by" ;  
TAG FOR is\_suggested\_by "Suggested by : " WITH  
ROLE 1 suggests "Suggests" ;

RELATIONSHIP to\_be\_utterance\_act;

TAG FOR is\_an\_utt\_act "Utterance Act : " WITH  
ROLE 1 is\_a\_sact\_1 "Speech Act" ;  
TAG FOR is\_a\_sact\_1 "Speech Act. : " WITH  
ROLE 1 is\_an\_utt\_act "Utterance Act" ;

RELATIONSHIP to\_be\_propositional\_act;

TAG FOR is\_a\_prop\_act "Propositional Act : " WITH  
ROLE 1 is\_a\_sact\_2 "Speech Act" ;  
TAG FOR is\_a\_sact\_2 "Speech Act.. : " WITH  
ROLE 1 is\_a\_prop\_act "Propositional Act" ;

RELATIONSHIP to\_be\_illocutionary\_act;

TAG FOR is\_an\_ill\_act "Illocutionary Act : " WITH  
ROLE 1 is\_a\_sact\_3 "Speech Act" ;  
TAG FOR is\_a\_sact\_3 "Speech Act... : " WITH  
ROLE 1 is\_an\_ill\_act "Illocutionary Act" ;

RELATIONSHIP to\_be\_perlocutionary\_act;

TAG FOR is\_a\_perl\_act "Perlocutionary Act : " WITH  
ROLE 1 is\_a\_sact\_4 "Speech Act" ;  
TAG FOR is\_a\_sact\_4 "Speech Act.... : " WITH  
ROLE 1 is\_a\_perl\_act "Perlocutionary Act" ;

RELATIONSHIP to\_be\_elementary\_ill\_act;

TAG FOR is\_an\_elem\_ill\_act "Elementary Illoc. Act : " WITH  
ROLE 1 is\_an\_ill\_act\_1 "Illocutionary Act" ;  
TAG FOR is\_an\_ill\_act\_1 "Illocutionary Act. : " WITH  
ROLE 1 is\_an\_elem\_ill\_act "Elementary Illoc. Act" ;

RELATIONSHIP to\_be\_complex\_ill\_act;

TAG FOR is\_a\_compl\_ill\_act "Complex Illocutionary Act : " WITH  
ROLE 1 is\_an\_ill\_act\_2 "Illocutionary Act" ;  
TAG FOR is\_an\_ill\_act\_2 "Illocutionary Act.. : " WITH

ROLE 1 is\_a\_compl\_ill\_act "Complex Illocutionary Act" ;

RELATIONSHIP to\_determine;

TAG FOR determines "Determines : " WITH  
 ROLE 1 implies "Implies" ,  
 ROLE 2 is\_determined\_by "Determined by" ;  
 TAG FOR implies "Implies : " WITH  
 ROLE 1 determines "Determines" ,  
 ROLE 2 is\_determined\_by "Determined by" ;  
 TAG FOR is\_determined\_by "Determined by : " WITH  
 ROLE 1 implies "Implies" ,  
 ROLE 2 determines "Determines" ;

RELATIONSHIP to\_represent\_elem\_ill\_act;

TAG FOR rep\_elem\_ill\_act "Represents El. Ill. Act : " WITH  
 ROLE 1 is\_rep\_by\_per\_act "Represented by Per. Act" ;  
 TAG FOR is\_rep\_by\_per\_act "Represented by Per. Act : " WITH  
 ROLE 1 rep\_elem\_ill\_act "Represents El. Ill. Act" ;

RELATIONSHIP to\_represent\_proposit\_act;

TAG FOR rep\_prop\_act "Represents Prop. Act : " WITH  
 ROLE 1 is\_rep\_by\_el\_ill\_act "Represented by El. Ill. Act" ;  
 TAG FOR is\_rep\_by\_el\_ill\_act "Represented by El. Ill. Act : " WITH  
 ROLE 1 rep\_prop\_act "Represents Prop. Act" ;

RELATIONSHIP to\_represent\_utterance\_act;

TAG FOR rep\_utt\_act "Represents Utterance Act : " WITH  
 ROLE 1 is\_rep\_by\_prop\_act "Represented by Prop. Act" ;  
 TAG FOR is\_rep\_by\_prop\_act "Represented by Prop. Act : " WITH  
 ROLE 1 rep\_utt\_act "Represents Utterance Act" ;

RELATIONSHIP to\_compose\_c\_and\_c;

TAG FOR comp\_c\_and\_c "Composes with another Comp. : " WITH  
 ROLE 1 is\_comp\_by\_c\_and\_c "Composed by 2 Complex" ;  
 TAG FOR is\_comp\_by\_c\_and\_c "Composed by 2 Complex : " WITH  
 ROLE 1 comp\_c\_and\_c "Composes with another Comp." ;

RELATIONSHIP to\_compose\_e\_and\_c;

TAG FOR comp1\_e\_and\_c "Composes with a Complex : " WITH  
 ROLE 1 comp2\_e\_and\_c "Composes with an Elem." ,

```

 ROLE 2 is_comp_by_e_and_c "Composed by El. and Compl." ;
TAG FOR comp2_e_and_c "Composes with an Elem. : " WITH
 ROLE 1 comp1_e_and_c "Composes with a Complex" ,
 ROLE 2 is_comp_by_e_and_c "Composed by El. and Compl." ;
TAG FOR is_comp_by_e_and_c "Composed by El. and Compl. : " WITH
 ROLE 1 comp1_e_and_c "Composes with a Complex" ,
 ROLE 2 comp2_e_and_c "Composes with an Elem." ;

```

RELATIONSHIP to\_compose\_e\_and\_e;

```

TAG FOR comp_e_and_e "Composes with another Elem. : " WITH
 ROLE 1 is_comp_by_e_and_e "Composed by 2 Elem." ;
TAG FOR is_comp_by_e_and_e "Composed by 2 Elem. : " WITH
 ROLE 1 comp_e_and_e "Composes with another Elem." ;

```

RELATIONSHIP to\_form\_not\_c;

```

TAG FOR forms_not_c "Forms with Complex Negation : " WITH
 ROLE 1 is_formed_by_not_c "Formed by not a Complex" ;
TAG FOR is_formed_by_not_c "Formed by not a Complex : " WITH
 ROLE 1 forms_not_c "Forms with Complex Negation" ;

```

RELATIONSHIP to\_form\_not\_e;

```

TAG FOR forms_not_e "Forms with Elem. Negation : " WITH
 ROLE 1 is_formed_by_not_e "Formed by not an Elem." ;
TAG FOR is_formed_by_not_e "Formed by not an Elem. : " WITH
 ROLE 1 forms_not_e "Forms with Elem. Negation" ;

```

RELATIONSHIP to\_implicate\_e\_p;

```

TAG FOR implicates_e_p "Implicates an Elem. : " WITH
 ROLE 1 is_implicated_by_e_p "Elem. implicated by Pred." ,
 ROLE 2 is_formed_by_e_p "Formed by Elem. and Pred." ;
TAG FOR is_implicated_by_e_p "Elem. implicated by Pred. : " WITH
 ROLE 1 implicates_e_p "Implicates an Elem." ,
 ROLE 2 is_formed_by_e_p "Formed by Elem. and Pred." ;
TAG FOR is_formed_by_e_p "Formed by Elem. and Pred. : " WITH
 ROLE 1 implicates_e_p "Implicates an Elem." ,
 ROLE 2 is_implicated_by_e_p "Elem. implicated by Pred" ;

```

RELATIONSHIP to\_implicate\_c\_p;

```

TAG FOR implicates_c_p "Implicates a Complex : " WITH
 ROLE 1 is_implicated_by_c_p "Complex implicated by Pred." ,

```

```
ROLE 2 is_formed_by_c_p "Formed by Complex and Pred." ;
TAG FOR is_implicated_by_c_p "Complex implicated by Pred. : " WITH
ROLE 1 implicates_c_p "Implicates a Complex" ,
ROLE 2 is_formed_by_c_p "Formed by Complex and Pred." ;
TAG FOR is_formed_by_c_p "Formed by Complex and Pred. : " WITH
ROLE 1 implicates_c_p "Implicates a Complex" ,
ROLE 2 is_implicated_by_c_p "Complex implicated by Pred." ;
```

```
/* PROPRIETES */
```

```
PROPERTY GOALS;
 TAG "Goals";
```

```
PROPERTY EFFECTS;
 TAG "Effects";
```

```
PROPERTY OBJECTIVES;
 TAG "Objectives";
```

```
PROPERTY AMBIGUITIES;
 TAG "Ambiguities";
```

```
PROPERTY LEVEL;
 TAG "Level";
```

```
PROPERTY NUMBER;
 TAG "Number";
```

```
PROPERTY CONTENT;
 TAG "Content";
```

```
PROPERTY ARGUMENTS;
 TAG "Arguments";
```

```
PROPERTY PLACE;
 TAG "Place";
```

```
PROPERTY TYPE;
 TAG "Type";
```

PROPERTY POSSIBLE\_WORLD;  
TAG "Possible World";

PROPERTY LIFE\_PERIOD;  
TAG "Life Period";

PROPERTY NATURE;  
TAG "Nature";

PROPERTY basic-name;  
TAG "Basic Name";

PROPERTY TIME\_FREQUENCY;  
TAG "Time / Frequency";

PROPERTY FORM;  
TAG "Form";

PROPERTY INTENTION;  
TAG "Intention";

PROPERTY ILLOCUTIONARY\_POINT;  
TAG "Illocutionary Point";

PROPERTY MODE\_OF\_ACHIEV\_OF\_ILL\_POINT;  
TAG "Mode of Achiev. of Ill. Point";

PROPERTY DEGREE\_OF\_STRENGTH\_ILL\_POINT;  
TAG "Degree of Strength Ill. Point";

PROPERTY PROPOSITIONAL\_CONDITIONS;  
TAG "Propositional Conditions";

PROPERTY PREPARATORY\_CONDITIONS;  
TAG "Preparatory Conditions";

PROPERTY SINCERITY\_CONDITIONS;  
TAG "Sincerity Conditions";



PROPERTY DEGREE\_OF\_STRENGTH\_SINC\_COND;  
TAG "Degree of Strength Sinc. Cond.";

PROPERTY PROPERTIES;  
TAG "Properties";

PROPERTY TARGET;  
TAG "Target";

PROPERTY PROBLEMS;  
TAG "Problems";

PROPERTY TASK;  
TAG "Task";

PROPERTY ILL\_FORCE;  
TAG "Illocutionary Force";

PROPERTY PROPOSITIONAL\_CONTENT;  
TAG "Propositional Content";

PROPERTY SUBGOAL;  
TAG "Subgoal";

PROPERTY PURPOSE;  
TAG "Purpose";

ANNEXE A5

LISTING DES INFORMATIONS ENTREES  
LORS DE L'INTRODUCTION DU CAS PETITPAS

LISTING DES INFORMATIONS ENTREES LORS DE  
L'INTRODUCTION DU CAS PETITPAS

Act : accounts\_card2\_sact

Governs

Involves

accounts\_card\_contract

customers\_accounts\_manag.

Act

accounts\_card2

Nature

internal

Basic Name

accounts\_card2\_sact .

Time / Frequency

at the same time of each sending note received

Act : accounts\_card\_sact

Governs

Involves

accounts\_card\_contract

customers\_accounts\_manag.

Succeeds

cheque\_encashment\_iact

cheque\_encashment\_sact

Act

accounts\_card

Preceeds

cheque3\_sact

Nature

internal

Basic Name

accounts\_card\_sact

Time / Frequency

for each cheque received

**Act : booking\_card\_sact**

Governs

booking\_card\_contract

Succeeds

waiting\_orders\_sact

Act

booking\_card

Preceeds

daily\_stock\_filing\_cards\_sact

Nature

external

Basic Name

booking\_card\_sact

Time / Frequency

after daily filing card receipt

Involves

preparation\_order

**Act : cheque2\_sact**

Allows

accepted\_cheque.

Governs

cheque\_contract

Act

cheque2

Nature

external

Basic Name

cheque2\_sact

Time / Frequency

Involves

order\_recording

accepted customer order and cheque as pay mode

**Act : cheque3\_sact**

Governs

cheque\_contract

Succeeds

accounts\_card\_sact

Act

cheque3

Nature

internal

Basic Name

cheque3\_sact

Time / Frequency

everyday (storage of received and validated cheques)

Involves

customers\_accounts\_manag.

**Act : cheque\_encashment\_iact**

Act

cheque\_encashment.

Performs

accountancy\_service\_agent

Preceeds

accounts\_card\_sact

Basic Name

cheque\_encashment\_iact

Time / Frequency

for each accounts card created (one card by cheque arrived)

**Act : cheque\_encashment\_sact**

Governs

cheque\_encashment\_contract

Act

cheque\_encashment

Preceeds

accounts\_card\_sact

Nature

external

Basic Name

cheque\_encashment\_sact

Time / Frequency

after the receipt of a validated cheque (and creation of a corresponding accounts card)

Act : cheque\_sact

Allows

cheque\_coming.

Governs

cheque\_contract

Act

cheque

Nature

external

Basic Name

cheque\_sact

Time / Frequency

when there is a cheque with the customer order

Act : customer\_order\_sact

Involves

customers\_accounts\_manag.

Involves

order\_recording

## Governs

order\_contract

## Act

customer\_order

## Nature

external

## Basic Name

customer\_order\_sact

## Time / Frequency

each receipted customer order

## Involves

order\_recording

## Act : daily\_stock\_filing\_cards\_sact

## Governs

daily\_stock\_filing\_cards\_contract

## Involves

stock\_state\_management

## Succeeds

booking\_card\_sact

## Simultaneous with

due/product\_sact

## Act

daily\_stock\_filing\_cards

## Nature

internal

## Basic Name

daily\_stock\_filing\_cards\_sact

## Time / Frequency

every day, for the delayed delivery (accumulation of the cards  
sended by the central stock service)

## Act : delivery\_paper2\_sact

**Governs**

delivery\_paper\_contract

Act

delivery\_paper2

Nature

external

Basic Name

delivery\_paper2\_sact

Time / Frequency

initial delivery

**Involves**

preparation\_order

**Act : delivery\_paper3\_sact****Governs**

delivery\_paper\_contract

Act

delivery\_paper3

Nature

internal

Basic Name

delivery\_paper3\_sact

Time / Frequency

initial or delayed delivery

**Involves**

preparation\_order

**Act : delivery\_paper4\_sact****Governs**

delivery\_paper\_contract

Succeeds

treated\_order\_sact

Act

**Involves**

preparation\_order



delivery\_paper4

Basic Name

delivery\_paper4\_sact

Time / Frequency

after a delayed delivery

Act : delivery\_paper\_sact

Governs

Involves

delivery\_paper\_contract

preparation\_order

Act

delivery\_paper

Nature

external

Basic Name

delivery\_paper\_sact

Time / Frequency

initial delivery

Act : due/product2\_sact

Governs

Involves

due/product\_contract

preparation\_order

Simultaneous with

waiting\_orders2\_sact

Act

due/product2

Nature

internal

Basic Name

due/product2\_sact

Time / Frequency

when there is a partial initial delivery

Act : due/product3\_sact

Governs

due/product\_contract

Act

due/product3

Nature

external

Basic Name

due/product3\_sact

Time / Frequency

when there is a partial delayed delivery

Involves

preparation\_order

Act : due/product\_sact

Governs

due/product\_contract

Simultaneous with

daily\_stock\_filing\_cards\_sact

Act

due/product

Nature

internal

Basic Name

due/product\_sact

Time / Frequency

after daily stock filing cards receipt

Involves

preparation\_order

**Act : exhausted2\_sact**

Allows

exhausted\_products.

Governs

exhausted\_contract

Simultaneous with

exhausted\_sact

Act

exhausted2

Nature

external

Basic Name

exhausted2\_sact

Time / Frequency

for the products which won't be able to be delivered

Involves

bill

**Act : exhausted3\_sact**

Governs

exhausted\_contract

Succeeds

repayment\_sact

repayments\_iact

Act

exhausted3

Nature

external

Basic Name

exhausted3\_sact

Time / Frequency

for the products which won't be delivered

Involves

customers\_accounts\_manag.

**Act : exhausted\_sact**

Allows

exhausted\_products.

Governs

exhausted\_contract

Involves

bill

Simultaneous with

exhausted2\_sact

Act

exhausted

Nature

external

Basic Name

exhausted\_sact

Time / Frequency

for the products which won't be able to be delivered

**Act : internal\_order\_paper7\_sact**

Governs

internal\_order\_paper\_contract

Involves

order\_recording

Selects

acceptance\_cust\_order.

Simultaneous with

internal\_order\_paper\_sact

Act

internal\_order\_paper7

Basic Name

internal\_order\_paper7\_sact

Time / Frequency

accepted customer order

**Act : internal\_order\_paper\_sact**

Governs

Involves

internal\_order\_paper\_contract

order\_recording

Simultaneous with

internal\_order\_paper7\_sact

Act

internal\_order\_paper

Nature

internal

Basic Name

internal\_order\_paper\_sact

Time / Frequency

accepted customer order

**Act : non\_delivery\_proof\_sact**

Allows

non\_deliver\_proof.

Governs

Involves

non\_delivery\_contract

bill

Act

non\_delivery\_proof

Nature

internal

Basic Name

non\_delivery\_proof\_sact

Time / Frequency

for the products which will ulteriorly be delivered

**Act : parcel\_constitution\_iact**

Act

parcel\_constitution.

Performs

order\_prepar\_service\_agent

Basic Name

parcel\_constitution\_iact

Time / Frequency

for each accepted order and for each selected delayed order

**Act : rejected\_order\_sact**

Governs

Involves

rejected\_order\_contract

order\_recording

Selects

acceptance\_cust\_order.

Act

rejected\_order

Nature

external

Basic Name

rejected\_order\_sact

Time / Frequency

erroneous order (the order or the eventual cheque)

**Act : repayments\_iact**

Act

repayments.

**Performs**

accountancy\_service\_agent

**Preceeds**

exhausted3\_sact

**Basic Name**

repayments\_iact

**Time / Frequency**

for each product which won't be delivered (initial or partial delivery)

**Act : repayment\_sact****Governs**

repayment\_contract

**Act**

repayment

**Preceeds**

exhausted3\_sact

**Nature**

external

**Basic Name**

repayment\_sact

**Time / Frequency**

after the receipt of a exhausted product note

**Involves**

customers\_accounts\_manag.

**Act : sending\_documents\_sact****Governs**

sending\_documents\_contract

**Act**

sending\_documents

**Nature****Involves**

bill

internal

Basic Name

sending\_documents\_sact

Time / Frequency

if there is a delivery

Act : sending\_iact

Act

sending.

Performs

sending\_service\_agent

Basic Name

sending\_iact

Time / Frequency

when receiving a parcel and the accompanying documents

Act : sending\_note2\_sact

Governs

Involves

sending\_note\_contract

bill

Act

sending\_note2

Basic Name

sending\_note2\_sact

Time / Frequency

for the products which must be delivered

Act : sending\_note3\_sact

Governs

Involves



sending\_note\_contract                      customers\_accounts\_manag.

Act

sending\_note3

Nature

external

Basic Name

sending\_note3\_sact

Time / Frequency

for each delivery

**Act : sending\_note4\_sact**

Governs

Involves

sending\_note\_contract

customers\_accounts\_manag.

Act

sending\_note4

Nature

internal

Basic Name

sending\_note4\_sact

Time / Frequency

after the receipt of a sending note

**Act : sending\_note5\_sact**

Governs

Involves

sending\_note\_contract

customers\_accounts\_manag.

Act

sending\_note5

Nature

internal

Basic Name

sending\_note5\_sact

Time / Frequency

for each repayment

Act : sending\_note\_sact

Governs

sending\_note\_contract

Act

sending\_note

Nature

external

Basic Name

sending\_note\_sact

Time / Frequency

for the products which must be delivered

Involves

bill

Act : treated\_order\_sact

Governs

treated\_order\_contract

Selects

treatment\_cust\_order.

Act

treated\_order

Preceeds

delivery\_paper4\_sact

Nature

internal

Basic Name

Involves

preparation\_order

treated\_order\_sact

Time / Frequency

when there will be no delayed delivery for the order

### Act : waiting\_orders2\_sact

Governs

Involves

waiting\_order\_contract

preparation\_order

Selects

treatment\_cust\_order.

Simultaneous with

due/product2\_sact

Act

waiting\_orders2

Basic Name

waiting\_orders2\_sact

Time / Frequency

when there will be a delayed delivery for the order

### Act : waiting\_orders\_sact

Governs

Involves

waiting\_order\_contract

preparation\_order

Act

waiting\_orders

Preceeds

booking\_card\_sact

Nature

internal

Basic Name

waiting\_orders\_sact

**Time / Frequency**

after booking

**Activity : bill****Forms**

bill\_service

Involved in

Governs

exhausted2\_sact

exhausted\_contract

Involved in

Governs

exhausted\_sact

exhausted\_contract

Involved in

Governs

non\_delivery\_proof\_sact

non\_delivery\_contract

Involved in

Governs

sending\_documents\_sact

sending\_documents\_contract

Involved in

Governs

sending\_note2\_sact

sending\_note\_contract

Involved in

Governs

sending\_note\_sact

sending\_note\_contract

**Goals**

To make the bill and the accompanying documents in connection with each order

**Basic Name**

bill

**Target**

- To set up and send correctly all the documents concerning a delivery

**Problems**

Contests with the customer

**Activity : customers\_accounts\_manag.**

## Forms

accountancy\_service

Involved in

Governs

accounts\_card2\_sact

accounts\_card\_contract

Involved in

Governs

accounts\_card\_sact

accounts\_card\_contract

Involved in

Governs

cheque3\_sact

cheque\_contract

Involved in

Governs

cheque\_encashment\_sact

cheque\_encashment\_contract

Involved in

Governs

exhausted3\_sact

exhausted\_contract

Involved in

Governs

repayment\_sact

repayment\_contract

Involved in

Governs

sending\_note3\_sact

sending\_note\_contract

Involved in

Governs

sending\_note4\_sact

sending\_note\_contract

Involved in

Governs

Involved in

Governs

sending\_note5\_sact

sending\_note\_contract

## Goals

To take charge of all the things which concern the customers accounts and the repayments to the customers

## Basic Name

customers\_accounts\_manag.

## Target

Update of the bill taking account of the deliveries done.  
To repay the customer for all the bought products and which will not be delivered

## Problems

Contests with the customer.

Working costs for the cheque treatment.  
 Payment in exchange for delivery not described

### Activity : order\_recording

#### Forms

order\_recording\_service

|                            |                               |
|----------------------------|-------------------------------|
| Involved in                | Governs                       |
| cheque2_sact               | cheque_contract               |
| Involved in                | Governs                       |
| cheque_sact                | cheque_contract               |
| Involved in                | Governs                       |
| customer_order_sact        | order_contract                |
| Involved in                | Governs                       |
| internal_order_paper7_sact | internal_order_paper_contract |
| Involved in                | Governs                       |
| internal_order_paper_sact  | internal_order_paper_contract |
| Involved in                | Governs                       |
| rejected_order_sact        | rejected_order_contract       |

#### Goals

Reception and control of the customers orders

#### Basic Name

order\_recording

#### Target

- Internal order paper creation for each accepted order
- Erroneous order rejection

#### Problems

- Rejection minimization for the phoned orders
- Contests with the customer

### Activity : preparation\_order

#### Forms

## preparation\_order\_service

|                      |                         |
|----------------------|-------------------------|
| Involved in          | Governs                 |
| booking_card_sact    | booking_card_contract   |
| Involved in          | Governs                 |
| delivery_paper2_sact | delivery_paper_contract |
| Involved in          | Governs                 |
| delivery_paper3_sact | delivery_paper_contract |
| Involved in          | Governs                 |
| delivery_paper4_sact | delivery_paper_contract |
| Involved in          | Governs                 |
| delivery_paper_sact  | delivery_paper_contract |
| Involved in          | Governs                 |
| due/product2_sact    | due/product_contract    |
| Involved in          | Governs                 |
| due/product3_sact    | due/product_contract    |
| Involved in          | Governs                 |
| due/product_sact     | due/product_contract    |
| Involved in          | Governs                 |
| Involved in          | Governs                 |
| treated_order_sact   | treated_order_contract  |
| Involved in          | Governs                 |
| waiting_orders2_sact | waiting_order_contract  |
| Involved in          | Governs                 |
| waiting_orders_sact  | waiting_order_contract  |

## Goals

- Making the parcel
- Management of the delayed orders

## Basic Name

preparation\_order

## Target

- To make a parcel for the sending service
- To take charge of the delayed orders
- To transmit the necessary informations to the other services

#### Problems

- Number of the delayed orders
- Unsatisfied delayed orders

### Activity : sending

#### Forms

sending\_service

#### Goals

Sending the parcel and accompanying documents to the customer

#### Basic Name

sending

#### Target

To respect the delivering engagement

#### Problems

- Contests with the customer
- Contests with the forwarding company

### Activity : stock\_state\_management

#### Involved in

#### Governs

daily\_stock\_filing\_cards\_sact    daily\_stock\_filing\_cards\_contract

#### Basic Name

stock\_state\_management

### Agent : accountancy\_service\_agent

#### Heard by

accounts\_card2  
cheque2  
cheque3  
exhausted3  
sending\_note3



sending\_note5

Held by

accountancy\_service

Performed by

cheque\_encashment\_iact  
repayments\_iact

Spoken by

accounts\_card  
cheque\_encashment  
repayment  
sending\_note4

Type

human, group

Basic Name

accountancy\_service\_agent

Agent : bill\_service\_agent

Heard by

delivery\_paper3

Held by

bill\_service

Spoken by

delivery\_paper4  
exhausted  
exhausted2  
exhausted3  
non\_delivery\_proof  
sending\_documents  
sending\_note  
sending\_note2  
sending\_note3

Type

human, group

Basic Name

bill\_service\_agent

**Agent : central\_stock\_service\_agent**

Heard by

booking\_card  
delivery\_paper  
due/product3

Type

human, group

Basic Name

central\_stock\_service\_agent

**Agent : clientele\_service\_agent**

Heard by

internal\_order\_paper7

Type

human, group

Basic Name

clientele\_service\_agent

**Agent : commercial\_department\_agent**

Heard by

delivery\_paper2

Type

human, group

Basic Name

commercial\_department\_agent

**Agent : customer\_agent**

Heard by

cheque\_encashment

exhausted  
rejected\_order  
repayment

Spoken by

cheque  
customer\_order

Type

human, individual

Basic Name

customer\_agent

Agent : information\_system\_agent

Heard by

accounts\_card  
due/product2  
sending\_note4  
treated\_order  
waiting\_orders2

Spoken by

accounts\_card2  
cheque3  
daily\_stock\_filing\_cards  
due/product  
sending\_note5  
waiting\_orders

Type

computer

Basic Name

information\_system\_agent

Agent : order\_prepar\_service\_agent

Heard by

daily\_stock\_filing\_cards  
delivery\_paper4  
due/product  
internal\_order\_paper  
waiting\_orders

Held by

preparation\_order\_service

Performed by

parcel\_constitution\_iact

Spoken by

booking\_card  
delivery\_paper  
delivery\_paper2  
delivery\_paper3  
due/product2  
due/product3  
treated\_order  
waiting\_orders2

Type

human, group

Basic Name

order\_prepar\_service\_agent

**Agent : order\_recording\_service\_agent**

Heard by

cheque  
customer\_order

Held by

order\_recording\_service

Spoken by

cheque2  
internal\_order\_paper  
internal\_order\_paper7  
rejected\_order

Type

human, group

Basic Name

order\_recording\_service\_agent

**Agent : sending\_service\_agent**

**Heard by**

non\_delivery\_proof  
 sending\_documents  
 sending\_note2

**Held by**

sending\_service

**Performed by**

sending\_iact

**Type**

human,group

**Basic Name**

sending\_service\_agent

**Commitment : booked\_quantities\_commit****Implies**

booking\_card\_eia

**Determines**

booking\_card\_if

**Content**

booked quantities commitment

**Type**

not weak

**Basic Name**

booked\_quantities\_commit

**Commitment : cheque\_commit****Implies**

accounts\_card\_eia

**Determines**

accounts\_card\_if

**Content**

cheque commitment

**Type**

weak

## Basic Name

cheque\_commit

## Commitment : customer\_delivery\_commit

|                                               |                       |
|-----------------------------------------------|-----------------------|
| Implies                                       | Determines            |
| delivery_paper2_eia                           | delivery_paper_if     |
| Implies                                       | Determines            |
| delivery_paper3_eia                           | delivery_paper_if     |
| Implies                                       | Determines            |
| delivery_paper4_eia                           | delivery_paper_if     |
| Implies                                       | Determines            |
| delivery_paper_eia                            | delivery_paper_if     |
| Implies                                       | Determines            |
| non_delivery_proof_eia                        | non_delivery_proof_if |
| Implies                                       | Determines            |
| sending_documents_eia                         | sending_documents_if  |
| Implies                                       | Determines            |
| sending_note2_eia                             | sending_note_if       |
| Implies                                       | Determines            |
| sending_note3_eia                             | sending_note_if       |
| Implies                                       | Determines            |
| sending_note4_eia                             | sending_note_if       |
| Implies                                       | Determines            |
| Implies                                       | Determines            |
| sending_note5_eia                             | sending_note_if       |
| Implies                                       | Determines            |
| sending_note_eia                              | sending_note_if       |
| Content                                       |                       |
| commitment to deliver product to the customer |                       |

## Basic Name

customer\_delivery\_commit

## Commitment : customer\_order\_commit

|                           |                         |
|---------------------------|-------------------------|
| Implies                   | Determines              |
| customer_order_eia        | customer_order_if       |
| Implies                   | Determines              |
| internal_order_paper7_eia | internal_order_paper_if |
| Implies                   | Determines              |
| internal_order_paper_eia  | internal_order_paper_if |
| Implies                   | Determines              |
| treated_order_eia         | treated_order_if        |
| Implies                   | Determines              |
| waiting_orders2_eia       | waiting_orders_if       |
| Implies                   | Determines              |
| waiting_orders_eia        | waiting_orders_if       |

## Content

commitment of the customer to order product

## Type

weak

## Basic Name

customer\_order\_commit

## Commitment : delivery1\_commit

|               |              |
|---------------|--------------|
| Implies       | Determines   |
| delivery1_eia | delivery1_if |

## Content

delivery commitment

## Type

not weak

Basic Name

delivery1\_commit

Commitment : due\_quantities\_commit

Implies

due/product2\_eia

Implies

due/product3\_eia

Implies

due/product\_eia

Content

due quantities commitment

Type

weak

Basic Name

due\_quantities\_commit

Determines

due/product\_if

Determines

due/product\_if

Determines

due/product\_if

Commitment : erroneous\_order\_commit

Implies

erroneous\_order\_eia

Type

not weak

Basic Name

erroneous\_order\_commit

Determines

erroneous\_order\_if

Commitment : exhausted\_product\_commit

Implies

exhausted2\_product\_eia

Determines

exhausted\_product\_if



|                          |                      |
|--------------------------|----------------------|
| Implies                  | Determines           |
| exhausted3_product_eia   | exhausted_product_if |
| Implies                  | Determines           |
| exhausted_product_eia    | exhausted_product_if |
| Type                     |                      |
| not weak                 |                      |
| Basic Name               |                      |
| exhausted_product_commit |                      |

#### Commitment : paid\_order\_commit

|                       |                      |
|-----------------------|----------------------|
| Implies               | Determines           |
| cheque_encashment_eia | cheque_encashment_if |
| Content               |                      |
| commitment to pay     |                      |
| Type                  |                      |
| not weak              |                      |
| Basic Name            |                      |
| paid_order_commit     |                      |

#### Commitment : payment\_commit

|             |            |
|-------------|------------|
| Implies     | Determines |
| cheque2_eia | cheque_if  |
| Implies     | Determines |
| cheque3_eia | cheque_if  |
| Implies     | Determines |
| cheque_eia  | cheque_if  |
| Implies     | Determines |
| payment_eia | payment_if |
| Content     |            |

commitment to pay

Type

not weak

Basic Name

payment\_commit

Commitment : repaid\_order\_commit

Implies

Determines

repayment\_eia

repayment\_if

Content

commitment to repay the order

Type

not weak

Basic Name

repaid\_order\_commit

Commitment : repayment1\_commit

Implies

Determines

repayment1\_eia

repayment1\_if

Content

commitment to repay

Type

not weak

Basic Name

repayment1\_commit

Commitment : repayment2\_commit

Implies

Determines

|                     |               |
|---------------------|---------------|
| repayment2_eia      | repayment2_if |
| Content             |               |
| commitment to repay |               |
| Type                |               |
| not weak            |               |
| Basic Name          |               |
| repayment2_commit   |               |

Commitment : repayment3\_commit

|                     |               |
|---------------------|---------------|
| Implies             | Determines    |
| repayment3_eia      | repayment3_if |
| Content             |               |
| commitment to repay |               |
| Type                |               |
| not weak            |               |
| Basic Name          |               |
| repayment3_commit   |               |

**Complex Illocutionary Act : customer\_order\_cia**

Composes with a Complex  
delivery1\_eia  
Complex Illocutionary Act  
customer\_order\_ia  
Basic Name  
customer\_order\_cia

Composes with an Elem.  
delivery\_cia

**Complex Illocutionary Act : delivery\_cia**

Composes with a Complex  
delivery1\_eia  
implicates an Elem.  
delivery\_pred.  
Basic Name  
delivery\_cia

Composed by El. and Compl.  
customer\_order\_cia  
Elem. implicated by Pred  
payment\_eia

**Complex Illocutionary Act : exhausted2\_cia**

Composes with another Elem.  
exhausted2\_product\_eia  
repayment2\_eia  
Complex Illocutionary Act  
exhausted2\_ia  
Basic Name  
exhausted2\_cia

**Complex Illocutionary Act : exhausted3\_cia**

Composes with another Elem.  
exhausted3\_product\_eia  
repayment3\_eia  
Complex Illocutionary Act

exhausted3\_ia

Basic Name

exhausted3\_cia

**Complex Illocutionary Act : exhausted\_cia**

Composes with another Elem.

exhausted\_product\_eia  
repayment1\_eia

Complex Illocutionary Act

exhausted\_ia

Basic Name

exhausted\_cia

**Complex Illocutionary Act : rejected\_order\_cia**

Composes with another Elem.

customer\_order\_eia  
erroneous\_order\_eia

Complex Illocutionary Act

rejected\_order\_ia

Basic Name

rejected\_order\_cia

**Conversation : cheque\_convers**

Described by

cheque\_encashment.

Enclosed in

cheque\_trig.\_pred.

Split in

customers\_accounts\_management

Triggers

cheque.

Basic Name

cheque\_convers

**Conversation : customer\_order\_convers**

Described by

parcel\_constitution.  
sending.

Enclosed in

accept\_cust\_order\_sel.\_pred.  
accepted\_cheque\_gate\_pred.  
cheque\_coming\_gate\_pred.  
customer\_order\_trig.\_pred.  
delivery\_pred.  
delivery\_sel.\_pred.  
exhausted\_products\_gate\_pred.  
non\_deliver\_proof\_gate\_pred.  
treatm\_cust\_order\_sel.\_pred.

Split in

customers\_orders\_management

Triggers

customer\_order.

Basic Name

customer\_order\_convers

**Conversation : exhausted\_product\_convers**

Described by

repayments.

Enclosed in

exhausted\_product\_trig.\_pred  
repayment\_sel.\_pred.

Split in

customers\_accounts\_management

Triggers

exhausted\_product.

Basic Name

exhausted\_product\_convers

**Conversation : sending\_note\_convers**

Enclosed in

sending\_note\_trig.\_pred.

Split in

customers\_accounts\_management

Basic Name

sending\_note\_convers

**Conversation : stock\_cards\_convers**

Enclosed in

delayed\_delivery\_sel.\_pred.  
due/product\_sel.\_pred.  
stock\_cards\_trig.\_pred.

Split in

customers\_orders\_management

Triggers

stock\_cards.

Basic Name

stock\_cards\_convers

**Discourse : small\_step**

Divides

customers\_accounts\_management  
customers\_orders\_management

Basic Name

small\_step

**Discourse Segment : cheque\_treatment\_segm**

Basic Name

cheque\_treatment\_segm

Subgoal

to encash a cheque

**Discourse Segment : customer\_delivery\_segm**

Grouped in

exhausted\_note\_stage

Basic Name

customer\_delivery\_segm

Subgoal

to update the customer accounts

**Discourse Segment : order\_treatment\_segm**

Grouped in

accept/rej\_cust\_order\_stage  
intern\_order\_paper\_arch\_stage

Basic Name

order\_treatment\_segm

Subgoal

Production of delivery paper from the customer order depending  
on stock availability

**Discourse Segment : products\_delivery\_segm**

Grouped in

delivery\_stage  
exhausted/sending\_stage  
parcel\_constitution\_stage



Basic Name

products\_delivery\_segm

Subgoal

Deliverable products routing to the customer

Discourse Type : customers\_accounts\_management

Divided in

small\_step

Splits in

cheque\_convers

exhausted\_product\_convers

sending\_note\_convers

Suggested by

exhausted\_note\_stage

Goals

Treatment of all about the customers payment

Effects

The received cheque is cashed and the customer is reimbursed if his order can't be (totally) respected

Objectives

To keep the customers accounts (payments and repayments)

Ambiguities

What happens for a payment in exchange for delivery ?

What about the contest with the customer ?

Level

base and secondary

Basic Name

customers\_accounts\_management

Discourse Type : customers\_orders\_management

Divided in

small\_step

Splits in

customer\_order\_convers  
stock\_cards\_convers

Suggested by

accept/rej\_cust\_order\_stage  
delivery\_stage  
exhausted/sending\_stage  
intern\_order\_paper\_arch\_stage  
parcel\_constitution\_stage

Goals

To make the treatment of the customers orders received and the  
(initial or delayed) delivery

Effects

The ordered products are delivered within the stock availability  
or the order is set aside and returned to the customer

Objectives

- To reduce as much as possible the delayed orders number and the lost sales
- To reduce the working costs of the cheque treatment
- To improve the service at the clientele

Ambiguities

What about the contest with the customer ?

Level

base and secondary

Basic Name

customers\_orders\_management

### Elementary Illocutionary Act : accounts\_card2\_eia

Elementary Illoc. Act

accounts\_card2\_ia

Represented by El. Ill. Act

accounts\_card2\_prop

Basic Name

accounts\_card2\_eia

**Elementary Illocutionary Act : accounts\_card\_eia**

Elementary Illoc. Act

accounts\_card\_ia

Determines

Determined by

accounts\_card\_if

cheque\_commit

Represented by El. Ill. Act

accounts\_card\_prop

Basic Name

accounts\_card\_eia

**Elementary Illocutionary Act : booking\_card\_eia**

Elementary Illoc. Act

booking\_card\_ia

Determines

Determined by

booking\_card\_if

booked\_quantities\_commit

Represented by El. Ill. Act

booking\_card\_prop

Basic Name

booking\_card\_eia

**Elementary Illocutionary Act : cheque2\_eia**

Elementary Illoc. Act

cheque2\_ia

Determines

Determined by

cheque\_if

payment\_commit

Represented by El. Ill. Act

cheque2\_prop

Basic Name

cheque2\_eia

**Elementary Illocutionary Act : cheque3\_eia**

Elementary Illoc. Act

cheque3\_ia

Determines

Determined by

cheque\_if

payment\_commit

Represented by El. Ill. Act

cheque3\_prop

Basic Name

cheque3\_eia

**Elementary Illocutionary Act : cheque\_eia**

Elementary Illoc. Act

cheque\_ia

Determines

Determined by

cheque\_if

payment\_commit

Represented by El. Ill. Act

cheque\_prop

Basic Name

cheque\_eia

**Elementary Illocutionary Act : cheque\_encashment\_eia**

Elementary Illoc. Act

cheque\_encashment\_ia

Determines

Determined by

cheque\_encashment\_if

paid\_order\_commit

Represented by El. Ill. Act

cheque\_encashment\_prop

Basic Name

cheque\_encashment\_eia

**Elementary Illocutionary Act : customer\_order\_eia**

Composed by 2 Elem.

rejected\_order\_cia

Determines

customer\_order\_if

Determined by

customer\_order\_commit

Basic Name

customer\_order\_eia

**Elementary Illocutionary Act : daily\_stock\_filing\_cards\_eia**

Elementary Illoc. Act

daily\_stock\_filing\_cards\_ia

Determines

daily\_stock\_filing\_cards\_if

Determined by

stock\_commit

Represented by El. Ill. Act

daily\_stock\_filing\_cards\_prop

Basic Name

daily\_stock\_filing\_cards\_eia

**Elementary Illocutionary Act : delivery1\_eia**

Composes with an Elem.

delivery\_cia

Composed by El. and Compl.

customer\_order\_cia

Determines

delivery1\_if

Determined by

delivery1\_commit

Basic Name

delivery1\_eia

**Elementary Illocutionary Act : delivery\_paper2\_eia**

Elementary Illoc. Act

delivery\_paper2\_ia

Determines

Determined by

delivery\_paper\_if

customer\_delivery\_commit

Represented by El. Ill. Act

delivery\_paper2\_prop

Basic Name

delivery\_paper2\_eia

**Elementary Illocutionary Act : delivery\_paper3\_eia**

Elementary Illoc. Act

delivery\_paper3\_ia

Determines

Determined by

delivery\_paper\_if

customer\_delivery\_commit

Represented by El. Ill. Act

delivery\_paper3\_prop

Basic Name

delivery\_paper3\_eia

**Elementary Illocutionary Act : delivery\_paper4\_eia**

Elementary Illoc. Act

delivery\_paper4\_ia

Determines

Determined by

delivery\_paper\_if

customer\_delivery\_commit

Represented by El. Ill. Act

delivery\_paper4\_prop

Basic Name

delivery\_paper4\_eia

**Elementary Illocutionary Act : delivery\_paper\_eia**

Elementary Illoc. Act

delivery\_paper\_ia

Determines

Determined by

delivery\_paper\_if

customer\_delivery\_commit

Represented by El. Ill. Act

delivery\_paper\_prop

Basic Name

delivery\_paper\_eia

**Elementary Illocutionary Act : due/product2\_eia**

Elementary Illoc. Act

due/product\_ia

Determines

Determined by

due/product\_if

due\_quantities\_commit

Represented by El. Ill. Act

due/product2\_prop

Basic Name

due/product2\_eia

**Elementary Illocutionary Act : due/product3\_eia**

Elementary Illoc. Act

due/product3\_ia

Determines

Determined by





Basic Name

exhausted2\_product\_eia

Elementary Illocutionary Act : exhausted3\_product\_eia

Composed by 2 Elem.

exhausted3\_cia

Determines

exhausted\_product\_if

Basic Name

exhausted3\_product\_eia

Determined by

exhausted\_product\_commit

Elementary Illocutionary Act : exhausted\_product\_eia

Composed by 2 Elem.

exhausted\_cia

Determines

exhausted\_product\_if

Basic Name

exhausted\_product\_eia

Determined by

exhausted\_product\_commit

Elementary Illocutionary Act : internal\_order\_paper7\_eia

Elementary Illoc. Act

internal\_order\_paper7\_ia

Determines

internal\_order\_paper\_if

Represented by El. Ill. Act

internal\_order\_paper7\_prop

Basic Name

internal\_order\_paper7\_eia

Determined by

customer\_order\_commit

**Elementary Illocutionary Act : internal\_order\_paper\_eia**

Elementary Illoc. Act

internal\_order\_paper\_ia

Determines

Determined by

internal\_order\_paper\_if

customer\_order\_commit

Represented by El. Ill. Act

internal\_order\_paper\_prop

Basic Name

internal\_order\_paper\_eia

**Elementary Illocutionary Act : non\_delivery\_proof\_eia**

Elementary Illoc. Act

non\_delivery\_proof\_ia

Determines

Determined by

non\_delivery\_proof\_if

customer\_delivery\_commit

Represented by El. Ill. Act

non\_delivery\_proof\_prop

Basic Name

non\_delivery\_proof\_eia

**Elementary Illocutionary Act : payment\_eia**

Implicates an Elem.

Formed by Elem. and Pred.

delivery\_pred.

delivery\_cia

Determines

Determined by

payment\_if

payment\_commit

Basic Name

payment\_eia

**Elementary Illocutionary Act : repayment1\_eia**

Composed by 2 Elem.

exhausted\_cia

Determines

Determined by

repayment1\_if

repayment1\_commit

Basic Name

repayment1\_eia

**Elementary Illocutionary Act : repayment2\_eia**

Composed by 2 Elem.

exhausted2\_cia

Determines

Determined by

repayment2\_if

repayment2\_commit

Basic Name

repayment2\_eia

**Elementary Illocutionary Act : repayment3\_eia**

Composed by 2 Elem.

exhausted3\_cia.

Determines

Determined by

repayment3\_if

repayment3\_commit

Basic Name

repayment3\_eia

**Elementary Illocutionary Act : repayment\_eia**

Elementary Illoc. Act

repayment\_ia

Determines

Determined by



sending\_note3\_ia

Determines

Determined by

sending\_note\_if

customer\_delivery\_commit

Represented by El. Ill. Act

sending\_note3\_prop

Basic Name

sending\_note3\_eia

#### Elementary Illocutionary Act : sending\_note4\_eia

Elementary Illoc. Act

sending\_note4\_ia

Determines

Determined by

sending\_note\_if

customer\_delivery\_commit

Represented by El. Ill. Act

sending\_note4\_prop

Basic Name

sending\_note4\_eia

#### Elementary Illocutionary Act : sending\_note5\_eia

Elementary Illoc. Act

sending\_note5\_ia

Determines

Determined by

sending\_note\_if

customer\_delivery\_commit

Represented by El. Ill. Act

sending\_note5\_prop

Basic Name

sending\_note5\_eia

#### Elementary Illocutionary Act : sending\_note\_eia

## Elementary Illoc. Act

sending\_note\_ia

Determines

Determined by

sending\_note\_if

customer\_delivery\_commit

Represented by El. Ill. Act

sending\_note\_prop

Basic Name

sending\_note\_eia

## Elementary Illocutionary Act : treated\_order\_eia

## Elementary Illoc. Act

treated\_order\_ia

Determines

Determined by

treated\_order\_if

customer\_order\_commit

Represented by El. Ill. Act

treated\_order\_prop

Basic Name

treated\_order\_eia

## Elementary Illocutionary Act : waiting\_orders2\_eia

## Elementary Illoc. Act

waiting\_orders2\_ia

Determines

Determined by

waiting\_orders\_if

customer\_order\_commit

Represented by El. Ill. Act

waiting\_orders2\_prop

Basic Name

waiting\_orders2\_eia

**Elementary Illocutionary Act : waiting\_orders\_eia**

Elementary Illoc. Act

waiting\_orders\_ia

Determines

Determined by

waiting\_orders\_if

customer\_order\_commit

Represented by El. Ill. Act

waiting\_orders\_prop

Basic Name

waiting\_orders\_eia

. Entity : product

Life Period

eternal

Basic Name

product

Properties

number

buyer service

gross stock

whole of dues

whole of ordered quantities

whole of received quantities

total in order

resource level

order point

order economic quantity

Gate Predicate : accepted\_cheque.

Allowed by

cheque2\_sact

Gate Predicate

accepted\_cheque\_gate\_pred.

Basic Name

accepted\_cheque.

Gate Predicate : cheque\_coming.

Allowed by

cheque\_sact

Gate Predicate

cheque\_coming\_gate\_pred.

Basic Name

cheque\_coming.

Gate Predicate : exhausted\_products.

Allowed by

exhausted2\_sact

exhausted\_sact

Gate Predicate

exhausted\_products\_gate\_pred.

Basic Name

exhausted\_products.

Gate Predicate : non\_deliver\_proof.

Allowed by

non\_delivery\_proof\_sact

Gate Predicate

non\_deliver\_proof\_gate\_pred.

Basic Name

non\_deliver\_proof.

Illocutionary Act : accounts\_card2\_ia



Illocutionary Act

accounts\_card2\_eia

Illocutionary Act

accounts\_card2

Basic Name

accounts\_card2\_ia

Illocutionary Act : accounts\_card\_ia

Illocutionary Act

accounts\_card\_eia

Illocutionary Act

accounts\_card

Basic Name

accounts\_card\_ia

Illocutionary Act : booking\_card\_ia

Illocutionary Act

booking\_card\_eia

Illocutionary Act

booking\_card

Basic Name

booking\_card\_ia

Illocutionary Act : cheque2\_ia

Illocutionary Act

cheque2\_eia

Illocutionary Act

cheque2

Basic Name

cheque2\_ia

Illocutionary Act : cheque3\_ia

Illocutionary Act

cheque3\_eia

Illocutionary Act

cheque3

Basic Name

cheque3\_ia

Illocutionary Act : cheque\_encashment\_ia

Illocutionary Act

cheque\_encashment\_eia

Illocutionary Act

cheque\_encashment

Basic Name

cheque\_encashment\_ia

Illocutionary Act : cheque\_ia

Illocutionary Act

cheque\_eia

Illocutionary Act

cheque

Basic Name

cheque\_ia

Illocutionary Act : customer\_order\_ia

Illocutionary Act

customer\_order\_cia

Illocutionary Act

customer\_order

Basic Name

customer\_order\_ia

Illocutionary Act : daily\_stock\_filing\_cards\_ia

Illocutionary Act

daily\_stock\_filing\_cards\_eia

Illocutionary Act

daily\_stock\_filing\_cards

Basic Name

daily\_stock\_filing\_cards\_ia

Illocutionary Act : delivery\_paper2\_ia

Illocutionary Act

delivery\_paper2\_eia

Illocutionary Act

delivery\_paper2

Basic Name

delivery\_paper2\_ia

Illocutionary Act : delivery\_paper3\_ia

Illocutionary Act

delivery\_paper3\_eia

Illocutionary Act

delivery\_paper3

Basic Name

delivery\_paper3\_ia

Illocutionary Act : delivery\_paper4\_ia

Illocutionary Act

delivery\_paper4\_eia

Illocutionary Act

delivery\_paper4

Basic Name

delivery\_paper4\_ia

Illocutionary Act : delivery\_paper\_ia

Illocutionary Act

delivery\_paper\_eia

Illocutionary Act

delivery\_paper

Basic Name

delivery\_paper\_ia

Illocutionary Act : due/product2\_ia

Illocutionary Act

due/product2

Basic Name

due/product2\_ia

Illocutionary Act : due/product3\_ia

Illocutionary Act

due/product3\_eia

Illocutionary Act

due/product3

Basic Name

due/product3\_ia

Illocutionary Act : due/product\_ia

Illocutionary Act

due/product2\_eia  
due/product\_eia

Illocutionary Act

due/product

Basic Name

due/product\_ia

Illocutionary Act : exhausted2\_ia

Illocutionary Act

exhausted2\_cia

Illocutionary Act

exhausted2

Basic Name

exhausted2\_ia

Illocutionary Act : exhausted3\_ia

Illocutionary Act

exhausted3\_cia

Illocutionary Act

exhausted3

Basic Name

exhausted3\_ia

**Illocutionary Act : exhausted\_ia**

Illocutionary Act

exhausted\_cia

Illocutionary Act

exhausted

Basic Name

exhausted\_ia

**Illocutionary Act : internal\_order\_paper7\_ia**

Illocutionary Act

internal\_order\_paper7\_eia

Illocutionary Act

internal\_order\_paper7

Basic Name

internal\_order\_paper7\_ia

**Illocutionary Act : internal\_order\_paper\_ia**

Illocutionary Act

internal\_order\_paper\_eia

Illocutionary Act

internal\_order\_paper

Basic Name

internal\_order\_paper\_ia

**Illocutionary Act : non\_delivery\_proof\_ia**

Illocutionary Act

non\_delivery\_proof\_eia

Illocutionary Act

non\_delivery\_proof

Basic Name

non\_delivery\_proof\_ia

Illocutionary Act : rejected\_order\_ia

Illocutionary Act

rejected\_order\_cia

Illocutionary Act

rejected\_order

Basic Name

rejected\_order\_ia

Illocutionary Act : repayment\_ia

Illocutionary Act

repayment\_eia

Illocutionary Act

repayment

Basic Name

repayment\_ia

Illocutionary Act : sending\_documents\_ia

Illocutionary Act

sending\_documents\_eia

Illocutionary Act

sending\_documents

Basic Name

sending\_documents\_ia

Illocutionary Act : sending\_note2\_ia

Illocutionary Act

sending\_note2\_eia

Illocutionary Act

sending\_note2

Basic Name

sending\_note2\_ia

Illocutionary Act : sending\_note3\_ia

Illocutionary Act

sending\_note3\_eia

Illocutionary Act

sending\_note3

Basic Name

sending\_note3\_ia

Illocutionary Act : sending\_note4\_ia

Illocutionary Act

sending\_note4\_eia

Illocutionary Act

sending\_note4

Basic Name

sending\_note4\_ia

Illocutionary Act : sending\_note5\_ia

Illocutionary Act

sending\_note5\_eia

Illocutionary Act

sending\_note5



Basic Name

sending\_note5\_ia

Illocutionary Act : sending\_note\_ia

Illocutionary Act

sending\_note\_eia

Illocutionary Act

sending\_note

Basic Name

sending\_note\_ia

Illocutionary Act : treated\_order\_ia

Illocutionary Act

treated\_order\_eia

Illocutionary Act

treated\_order

Basic Name

treated\_order\_ia

Illocutionary Act : waiting\_orders2\_ia

Illocutionary Act

waiting\_orders2\_eia

Illocutionary Act

waiting\_orders2

Basic Name

waiting\_orders2\_ia

Illocutionary Act : waiting\_orders\_ia

Illocutionary Act

waiting\_orders\_eia

Illocutionary Act

waiting\_orders

Basic Name

waiting\_orders\_ia

**Illocutionary Force : accounts\_card\_if**

Implies

Determined by

accounts\_card\_eia

cheque\_commit

Basic Name

accounts\_card\_if

Illocutionary Point

assertive

**Illocutionary Force : booking\_card\_if**

Implies

Determined by

booking\_card\_eia

booked\_quantities\_commit

Basic Name

booking\_card\_if

Illocutionary Point

declarative

**Illocutionary Force : cheque\_encashment\_if**

Implies

Determined by

cheque\_encashment\_eia

paid\_order\_commit

Basic Name

cheque\_encashment\_if

**Illocutionary Point****assertive****Illocutionary Force : cheque\_if**

|             |                |
|-------------|----------------|
| Implies     | Determined by  |
| cheque2_eia | payment_commit |
| Implies     | Determined by  |
| cheque3_eia | payment_commit |
| Implies     | Determined by  |
| cheque_eia  | payment_commit |

**Basic Name****cheque\_if****Illocutionary Point****assertive****Illocutionary Force : customer\_order\_if**

|                    |                       |
|--------------------|-----------------------|
| Implies            | Determined by         |
| customer_order_eia | customer_order_commit |

**Basic Name****customer\_order\_if****Illocutionary Point****commissive****Illocutionary Force : daily\_stock\_filing\_cards\_if**

|                              |               |
|------------------------------|---------------|
| Implies                      | Determined by |
| daily_stock_filing_cards_eia | stock_commit  |

**Basic Name****daily\_stock\_filing\_cards\_if****Illocutionary Point**

assertive

### Illocutionary Force : delivery1\_if

|                     |                  |
|---------------------|------------------|
| Implies             | Determined by    |
| delivery1_eia       | delivery1_commit |
| Basic Name          |                  |
| delivery1_if        |                  |
| Illocutionary Point |                  |
| directive           |                  |

### Illocutionary Force : delivery\_paper\_if

|                     |                          |
|---------------------|--------------------------|
| Implies             | Determined by            |
| delivery_paper2_eia | customer_delivery_commit |
| Implies             | Determined by            |
| delivery_paper3_eia | customer_delivery_commit |
| Implies             | Determined by            |
| delivery_paper4_eia | customer_delivery_commit |
| Implies             | Determined by            |
| delivery_paper_eia  | customer_delivery_commit |
| Basic Name          |                          |
| delivery_paper_if   |                          |
| Illocutionary Point |                          |
| assertive           |                          |

### Illocutionary Force : due/product\_if

|                  |                       |
|------------------|-----------------------|
| Implies          | Determined by         |
| due/product2_eia | due_quantities_commit |
| Implies          | Determined by         |

|                     |                       |
|---------------------|-----------------------|
| due/product3_eia    | due_quantities_commit |
| Implies             | Determined by         |
| due/product_eia     | due_quantities_commit |
| Basic Name          |                       |
| due/product_if      |                       |
| Illocutionary Point |                       |
| assertive           |                       |

Illocutionary Force : erroneous\_order\_if

|                     |                        |
|---------------------|------------------------|
| Implies             | Determined by          |
| erroneous_order_eia | erroneous_order_commit |
| Basic Name          |                        |
| erroneous_order_if  |                        |
| Illocutionary Point |                        |
| assertive           |                        |

Illocutionary Force : exhausted\_product\_if

|                        |                          |
|------------------------|--------------------------|
| Implies                | Determined by            |
| exhausted2_product_eia | exhausted_product_commit |
| Implies                | Determined by            |
| exhausted3_product_eia | exhausted_product_commit |
| Implies                | Determined by            |
| exhausted_product_eia  | exhausted_product_commit |
| Basic Name             |                          |
| exhausted_product_if   |                          |
| Illocutionary Point    |                          |
| assertive              |                          |

Illocutionary Force : internal\_order\_paper\_if

|                           |                       |
|---------------------------|-----------------------|
| Implies                   | Determined by         |
| internal_order_paper7_eia | customer_order_commit |
| Implies                   | Determined by         |
| internal_order_paper_eia  | customer_order_commit |
| Basic Name                |                       |
| internal_order_paper_if   |                       |
| Illocutionary Point       |                       |
| assertive                 |                       |

Illocutionary Force : non\_delivery\_proof\_if

|                        |                          |
|------------------------|--------------------------|
| Implies                | Determined by            |
| non_delivery_proof_eia | customer_delivery_commit |
| Basic Name             |                          |
| non_delivery_proof_if  |                          |
| Illocutionary Point    |                          |
| assertive              |                          |

Illocutionary Force : payment\_if

|                     |                |
|---------------------|----------------|
| Implies             | Determined by  |
| payment_eia         | payment_commit |
| Basic Name          |                |
| payment_if          |                |
| Illocutionary Point |                |
| commissive          |                |

Illocutionary Force : repayment1\_if

|                |                   |
|----------------|-------------------|
| Implies        | Determined by     |
| repayment1_eia | repayment1_commit |

Basic Name

repayment1\_if

Illocutionary Point

commissive

Illocutionary Force : repayment2\_if

Implies

Determined by

repayment2\_eia

repayment2\_commit

Basic Name

repayment2\_if

Illocutionary Point

commissive

Illocutionary Force : repayment3\_if

Implies

Determined by

repayment3\_eia

repayment3\_commit

Basic Name

repayment3\_if

Illocutionary Point

commissive

Illocutionary Force : repayment\_if

Implies

Determined by

repayment\_eia

repaid\_order\_commit

Basic Name

repayment\_if

Illocutionary Point

assertive

**Illocutionary Force : sending\_documents\_if**

|                       |                          |
|-----------------------|--------------------------|
| Implies               | Determined by            |
| sending_documents_eia | customer_delivery_commit |
| Basic Name            |                          |
| sending_documents_if  |                          |
| Illocutionary Point   |                          |
| assertive             |                          |

**Illocutionary Force : sending\_note\_if**

|                     |                          |
|---------------------|--------------------------|
| Implies             | Determined by            |
| sending_note2_eia   | customer_delivery_commit |
| Implies             | Determined by            |
| sending_note3_eia   | customer_delivery_commit |
| Implies             | Determined by            |
| sending_note4_eia   | customer_delivery_commit |
| Implies             | Determined by            |
| sending_note5_eia   | customer_delivery_commit |
| Implies             | Determined by            |
| sending_note_eia    | customer_delivery_commit |
| Basic Name          |                          |
| sending_note_if     |                          |
| Illocutionary Point |                          |
| assertive           |                          |

**Illocutionary Force : treated\_order\_if**

|                   |                       |
|-------------------|-----------------------|
| Implies           | Determined by         |
| treated_order_eia | customer_order_commit |
| Basic Name        |                       |



treated\_order\_if

Illocutionary Point

assertive

Illocutionary Force : waiting\_orders\_if

Implies

Determined by

waiting\_orders2\_eia

customer\_order\_commit

Implies

Determined by

waiting\_orders\_eia

customer\_order\_commit

Basic Name

waiting\_orders\_if

Illocutionary Point

assertive

Instrumental Act : cheque\_encashment.

Instrumental Act

cheque\_encashment\_iact

Describes Instrumental Acts

cheque\_convers

Goals

to cash the cheque

Arguments

cheque

Basic Name

cheque\_encashment.

Instrumental Act : parcel\_constitution.

Instrumental Act

parcel\_constitution\_iact

**Describes Instrumental Acts**

customer\_order\_convers

**Goals**

to provide to the sending service the products to be delivered

**Arguments**

products to be delivered

**Basic Name**

parcel\_constitution.

**Instrumental Act : repayments.****Instrumental Act**

repayments\_iact

**Describes Instrumental Acts**

exhausted\_product\_convers

**Goals**

to repay the customer for what he has paid and not received

**Arguments**

paid product

**Basic Name**

repayments.

**Instrumental Act : sending.****Instrumental Act**

sending\_iact

**Describes Instrumental Acts**

customer\_order\_convers

**Goals**

parcel and documents sending

**Arguments**

products to be delivered

Basic Name

sending.

**Organisational Arrangement : accounts\_card\_contract**

Involved in

accounts\_card2\_sact

Involved in

accounts\_card\_sact

Basic Name

accounts\_card\_contract

Involves

customers\_accounts\_manag.

Involves

customers\_accounts\_manag.

**Organisational Arrangement : booking\_card\_contract**

Involved in

booking\_card\_sact

Basic Name

booking\_card\_contract

Involves

preparation\_order

**Organisational Arrangement : cheque\_contract**

Involved in

cheque2\_sact

Involved in

cheque3\_sact

Involved in

cheque\_sact

Basic Name

cheque\_contract

Involves

order\_recording

Involves

customers\_accounts\_manag.

Involves

order\_recording

**Organisational Arrangement : cheque\_encashment\_contract**

|                            |                           |
|----------------------------|---------------------------|
| Involved in                | Involves                  |
| cheque_encashment_sact     | customers_accounts_manag. |
| Basic Name                 |                           |
| cheque_encashment_contract |                           |

**Organisational Arrangement : daily\_stock\_filing\_cards\_contract**

|                                |                        |
|--------------------------------|------------------------|
| Involved in                    | Involves               |
| daily_stock_filing_cards_sact  | stock_state_management |
| Basic Name                     |                        |
| daily_stock_filing_cards_contr |                        |

**Organisational Arrangement : delivery\_paper\_contract**

|                         |                   |
|-------------------------|-------------------|
| Involved in             | Involves          |
| delivery_paper2_sact    | preparation_order |
| Involved in             | Involves          |
| delivery_paper3_sact    | preparation_order |
| Involved in             | Involves          |
| delivery_paper4_sact    | preparation_order |
| Involved in             | Involves          |
| delivery_paper_sact     | preparation_order |
| Basic Name              |                   |
| delivery_paper_contract |                   |

**Organisational Arrangement : due/product\_contract**

|                   |                   |
|-------------------|-------------------|
| Involved in       | Involves          |
| due/product2_sact | preparation_order |
| Involved in       | Involves          |

|                      |                   |
|----------------------|-------------------|
| due/product3_sact    | preparation_order |
| Involved in          | Involves          |
| due/product_sact     | preparation_order |
| Basic Name           |                   |
| due/product_contract |                   |

**Organisational Arrangement : exhausted\_contract**

|                    |                           |
|--------------------|---------------------------|
| Involved in        | Involves                  |
| exhausted2_sact    | bill                      |
| Involved in        | Involves                  |
| exhausted3_sact    | customers_accounts_manag. |
| Involved in        | Involves                  |
| exhausted_sact     | bill                      |
| Basic Name         |                           |
| exhausted_contract |                           |

**Organisational Arrangement : internal\_order\_paper\_contract**

|                               |                 |
|-------------------------------|-----------------|
| Involved in                   | Involves        |
| internal_order_paper7_sact    | order_recording |
| Involved in                   | Involves        |
| internal_order_paper_sact     | order_recording |
| Basic Name                    |                 |
| internal_order_paper_contract |                 |

**Organisational Arrangement : non\_delivery\_contract**

|                         |          |
|-------------------------|----------|
| Involved in             | Involves |
| non_delivery_proof_sact | bill     |
| Basic Name              |          |
| non_delivery_contract   |          |

**Organisational Arrangement : order\_contract**

|                     |                 |
|---------------------|-----------------|
| Involved in         | Involves        |
| customer_order_sact | order_recording |
| Basic Name          |                 |
| order_contract      |                 |

**Organisational Arrangement : rejected\_order\_contract**

|                         |                 |
|-------------------------|-----------------|
| Involved in             | Involves        |
| rejected_order_sact     | order_recording |
| Basic Name              |                 |
| rejected_order_contract |                 |

**Organisational Arrangement : repayment\_contract**

|                    |                           |
|--------------------|---------------------------|
| Involved in        | Involves                  |
| repayment_sact     | customers_accounts_manag. |
| Basic Name         |                           |
| repayment_contract |                           |

**Organisational Arrangement : sending\_documents\_contract**

|                            |          |
|----------------------------|----------|
| Involved in                | Involves |
| sending_documents_sact     | bill     |
| Basic Name                 |          |
| sending_documents_contract |          |

**Organisational Arrangement : sending\_note\_contract**

|             |          |
|-------------|----------|
| Involved in | Involves |
|-------------|----------|

|                       |                           |
|-----------------------|---------------------------|
| sending_note2_sact    | bill                      |
| Involved in           | Involves                  |
| sending_note3_sact    | customers_accounts_manag. |
| Involved in           | Involves                  |
| sending_note4_sact    | customers_accounts_manag. |
| Involved in           | Involves                  |
| sending_note5_sact    | customers_accounts_manag. |
| Involved in           | Involves                  |
| sending_note_sact     | bill                      |
| Basic Name            |                           |
| sending_note_contract |                           |

Organisational Arrangement : treated\_order\_contract

|                        |                   |
|------------------------|-------------------|
| Involved in            | Involves          |
| treated_order_sact     | preparation_order |
| Basic Name             |                   |
| treated_order_contract |                   |

Organisational Arrangement : waiting\_order\_contract

|                        |                   |
|------------------------|-------------------|
| Involved in            | Involves          |
| waiting_orders2_sact   | preparation_order |
| Involved in            | Involves          |
| waiting_orders_sact    | preparation_order |
| Basic Name             |                   |
| waiting_order_contract |                   |

Position : accountancy\_service

Formed by  
customers\_accounts\_manag.

holds

accountancy\_service\_agent

Basic Name

accountancy\_service

Task

to manage the treasury (cheque encashment and repayment)

Position : bill\_service

Formed by

bill

holds

bill\_service\_agent

Basic Name

bill\_service

Task

To create bills, sending documents and eventual non delivery proofs

Position : order\_recording\_service

Formed by

order\_recording

holds

order\_recording\_service\_agent

Basic Name

order\_recording\_service

Task

to control the received orders, to reject the erroneous orders and to create, for each accepted order, the internal order paper

Position : preparation\_order\_service



Formed by

preparation\_order

holds

order\_prepar\_service\_agent

Basic Name

preparation\_order\_service

Task

to constitute the parcels, to pack them up and to weigh them

Position : sending\_service

Formed by

sending

holds

sending\_service\_agent

Basic Name

sending\_service

Task

to organize the parcel routing to the customer and the contacts with the forwarding companies

Predicate : accepted\_cheque\_gate\_pred.

Encloses

customer\_order\_convers

Predicate

accepted\_cheque.

Number

G2

Content

If the order including a cheque is accepted (order and cheque valid)

Basic Name

accepted\_cheque\_gate\_pred.

Predicate : accept\_cust\_order\_sel.\_pred.

Encloses

customer\_order\_convers

Predicate

acceptance\_cust\_order.

Number

S1

Content

The order informations are valid (after eventual correctings) and the eventual cheque is valid.

XOR

The order can't be validate (error in the order or the cheque)

Basic Name

accept\_cust\_order\_sel.\_pred.

Predicate : cheque\_coming\_gate\_pred.

Encloses

customer\_order\_convers

Predicate

cheque\_coming.

Number

G1

Content

If there is a cheque with the order

Basic Name

cheque\_coming\_gate\_pred.

Predicate : cheque\_trig.\_pred.

Encloses

cheque\_convers

Predicate

cheque.

Number

T3

Content

everyday

Basic Name

cheque\_trig.\_pred.

Predicate : customer\_order\_trig.\_pred.

Encloses

customer\_order\_convers

Predicate

customer\_order.

Number

T1

Content

At each customer order

Basic Name

customer\_order\_trig.\_pred.

Predicate : delayed\_delivery\_sel.\_pred.

Encloses

stock\_cards\_convers

Predicate

delayed\_delivery.

Number

S4

Content

The delayed order

is completed and/or the time-limit for delivery is achieved.

XOR

isn't completed and the time-limit for delivery isn't achieved

Basic Name

delayed\_delivery\_sel.\_pred.

Predicate : delivery\_pred.

Encloses

customer\_order\_convers

Elem. implicated by Pred.

Formed by Elem. and Pred.

payment\_eia

delivery\_cia

Basic Name

delivery\_pred.

Predicate : delivery\_sel.\_pred.

Encloses

customer\_order\_convers

Predicate

delivery.

Number

S3

Content

There is delivery (even partial).

XOR

All the products to be delivered are exhausted

Basic Name

delivery\_sel.\_pred.

Predicate : due/product\_sel.\_pred.

Encloses

stock\_cards\_convers

Predicate

due/product.

Number

S5

Content

The delayed order

is completed

XOR

isn't completed and the delayed delivery time-limit is achieved

Basic Name

due/product\_sel.\_pred.

Predicate : exhausted\_products\_gate\_pred.

Encloses

customer\_order\_convers

Predicate

exhausted\_products.

Number

G3

## Content

If there are products exhausted (e.g. they will not be delivered)

Basic Name

exhausted\_products\_gate\_pred.

Predicate : exhausted\_product\_trig.\_pred

Encloses

exhausted\_product\_convers

Predicate

exhausted\_product.

Number

T5

Content

for each "exhausted" note received

Basic Name

exhausted\_product\_trig.\_pred

Predicate : non\_deliver\_proof\_gate\_pred.

Encloses

customer\_order\_convers

Predicate

non\_deliver\_proof.

Number

G4

Content

If the initial delivery is incomplete

Basic Name

non\_deliver\_proof\_gate\_pred.

Predicate : repayment\_sel.\_pred.

Encloses

exhausted\_product\_convers

Predicate

repayment.

Number

S6

Content

A sending note has already been sent for that order

XOR

There is'nt sending note for that order

Basic Name

repayment\_sel.\_pred.

Predicate : sending\_note\_trig.\_pred.

Encloses

sending\_note\_convers

Predicate

sending\_note.

Number

T4

Content

for each sending note received

Basic Name

sending\_note\_trig.\_pred.

Predicate : stock\_cards\_trig.\_pred.

Encloses

stock\_cards\_convers

Predicate

stock\_cards.

Number

T2

Content

Everyday

Basic Name

stock\_cards\_trig.\_pred.

Predicate : treatm\_cust\_order\_sel.\_pred.

Encloses

customer\_order\_convers

Predicate

treatment\_cust\_order.

Number

S2

Content

The initial order can be totally respected or the delayed order is treated

XOR

There will be a delayed delivery for that order

Basic Name

treatm\_cust\_order\_sel.\_pred.

Propositional Act : accounts\_card2\_prop

Represents Prop. Act

accounts\_card2\_eia

Represented by Prop. Act

accounts\_card2\_ua



**Propositional Act**

accounts\_card2

Content

cheque

Basic Name

accounts\_card2\_prop

**Propositional Act : accounts\_card\_prop**

Represents Prop. Act

accounts\_card\_eia

Represented by Prop. Act

accounts\_card\_ua

Propositional Act

accounts\_card

Content

cheque

Basic Name

accounts\_card\_prop

**Propositional Act : booking\_card\_prop**

Represents Prop. Act

booking\_card\_eia

Represented by Prop. Act

booking\_card\_ua

Propositional Act

booking\_card

Content

booked quantities

Basic Name

booking\_card\_prop

Propositional Act : cheque2\_prop

Represents Prop. Act

cheque2\_eia

Represented by Prop. Act

cheque2\_ua

Propositional Act

cheque2

Content

payment

Basic Name

cheque2\_prop

Propositional Act : cheque3\_prop

Represents Prop. Act

cheque3\_eia

Represented by Prop. Act

cheque3\_ua

Propositional Act

cheque3

Content

payment

Basic Name

cheque3\_prop

Propositional Act : cheque\_encashment\_prop

Represents Prop. Act

cheque\_encashment\_eia

Represented by Prop. Act

cheque\_encashment\_ua

Propositional Act

cheque\_encashment

Content

paid order

Basic Name

cheque\_encashment\_prop

#### Propositional Act : cheque\_prop

Represents Prop. Act

cheque\_eia

Represented by Prop. Act

cheque\_ua

Propositional Act

cheque

Content

payment

Basic Name

cheque\_prop

#### Propositional Act : customer\_order\_prop

Represented by Prop. Act

customer\_order\_ua

Propositional Act

customer\_order

Basic Name

customer\_order\_prop

**Propositional Act : daily\_stock\_filing\_cards\_prop**

Represents Prop. Act

daily\_stock\_filing\_cards\_eia

Represented by Prop. Act

daily\_stock\_filing\_cards\_ua

Propositional Act

daily\_stock\_filing\_cards

Content

stock

Basic Name

daily\_stock\_filing\_cards\_prop

**Propositional Act : delivery\_paper2\_prop**

Represents Prop. Act

delivery\_paper2\_eia

Represented by Prop. Act

delivery\_paper2\_ua

Propositional Act

delivery\_paper2

Content

customer delivery

Basic Name

delivery\_paper2\_prop

**Propositional Act : delivery\_paper3\_prop**

Represents Prop. Act

delivery\_paper3\_eia

Represented by Prop. Act

delivery\_paper3\_ua

Propositional Act

delivery\_paper3

Content

customer delivery

Basic Name

delivery\_paper3\_prop

**Propositional Act : delivery\_paper4\_prop**

Represents Prop. Act

delivery\_paper4\_eia

Represented by Prop. Act

delivery\_paper4\_ua

Propositional Act

delivery\_paper4

Content

customer delivery

Basic Name

delivery\_paper4\_prop

**Propositional Act : delivery\_paper\_prop**

Represents Prop. Act

delivery\_paper\_eia

Represented by Prop. Act

delivery\_paper\_ua

Propositional Act

delivery\_paper

Content

customer delivery

Basic Name

delivery\_paper\_prop

Propositional Act : due/product2\_prop

Represents Prop. Act

due/product2\_eia

Represented by Prop. Act

due/product2\_ua

Propositional Act

due/product2

Content

due quantities

Basic Name

due/product2\_prop

Propositional Act : due/product3\_prop

Represents Prop. Act

due/product3\_eia

Represented by Prop. Act

due/product3\_ua

Propositional Act

due/product3

Content

due quantities

Basic Name

due/product3\_prop

Propositional Act : due/product\_prop

Represents Prop. Act

due/product\_eia

Represented by Prop. Act

due/product\_ua

Propositional Act

due/product

Content

due quantities

Basic Name

due/product\_prop

Propositional Act : exhausted2\_prop

Represented by Prop. Act

exhausted2\_ua

Propositional Act

exhausted2

Basic Name

exhausted2\_prop

Propositional Act : exhausted3\_prop

Represented by Prop. Act

exhausted3\_ua

Propositional Act

exhausted3

Basic Name

exhausted3\_prop

Propositional Act : exhausted\_prop

Represented by Prop. Act

exhausted\_ua

Propositional Act

exhausted

Basic Name

exhausted\_prop

Propositional Act : internal\_order\_paper7\_prop

Represents Prop. Act

internal\_order\_paper7\_eia

Represented by Prop. Act

internal\_order\_paper7\_ua

Propositional Act

internal\_order\_paper7

Content

customer order

Basic Name

internal\_order\_paper7\_prop

Propositional Act : internal\_order\_paper\_prop

Represents Prop. Act

internal\_order\_paper\_eia

Represented by Prop. Act

internal\_order\_paper\_ua

Propositional Act

internal\_order\_paper

Content

customer order

Basic Name

internal\_order\_paper\_prop



**Propositional Act : non\_delivery\_proof\_prop**

Represents Prop. Act

non\_delivery\_proof\_eia

Represented by Prop. Act

non\_delivery\_proof\_ua

Propositional Act

non\_delivery\_proof

Content

customer delivery

Basic Name

non\_delivery\_proof\_prop

**Propositional Act : rejected\_order\_prop**

Represented by Prop. Act

rejected\_order\_ua

Propositional Act

rejected\_order

Basic Name

rejected\_order\_prop

**Propositional Act : repayment\_prop**

Represents Prop. Act

repayment\_eia

Represented by Prop. Act

repayment\_ua

Propositional Act

repayment

Content

repaid order

Basic Name

repayment\_prop

**Propositional Act : sending\_documents\_prop**

Represents Prop. Act

sending\_documents\_eia

Represented by Prop. Act

sending\_documents\_ua

Propositional Act

sending\_documents

Content

customer delivery

Basic Name

sending\_documents\_prop

**Propositional Act : sending\_note2\_prop**

Represents Prop. Act

sending\_note2\_eia

Represented by Prop. Act

sending\_note2\_ua

Propositional Act

sending\_note2

Content

customer delivery

Basic Name

sending\_note2\_prop

**Propositional Act : sending\_note3\_prop**

Represents Prop. Act

sending\_note3\_eia

Represented by Prop. Act

sending\_note3\_ua

Propositional Act

sending\_note3

Content

customer delivery

Basic Name

sending\_note3\_prop

Propositional Act : sending\_note4\_prop

Represents Prop. Act

sending\_note4\_eia

Represented by Prop. Act

sending\_note4\_ua

Propositional Act

sending\_note4

Content

customer delivery

Basic Name

sending\_note4\_prop

Propositional Act : sending\_note5\_prop

Represents Prop. Act

sending\_note5\_eia

Represented by Prop. Act

sending\_note5\_ua

Propositional Act

sending\_note5

Content

customer delivery

Basic Name

sending\_note5\_prop

**Propositional Act : sending\_note\_prop**

Represents Prop. Act

sending\_note\_eia

Represented by Prop. Act

sending\_note\_ua

Propositional Act

sending\_note

Content

customer delivery

Basic Name

sending\_note\_prop

**Propositional Act : treated\_order\_prop**

Represents Prop. Act

treated\_order\_eia

Represented by Prop. Act

treated\_order\_ua

Propositional Act

treated\_order

Content

customer order

Basic Name

treated\_order\_prop

**Propositional Act : waiting\_orders2\_prop**

Represents Prop. Act

waiting\_orders2\_eia

Represented by Prop. Act

waiting\_orders2\_ua

Propositional Act

waiting\_orders2

Content

customer order

Basic Name

waiting\_orders2\_prop

**Propositional Act : waiting\_orders\_prop**

Represents Prop. Act

waiting\_orders\_eia

Represented by Prop. Act

waiting\_orders\_ua

Propositional Act

waiting\_orders

Content

customer order

Basic Name

waiting\_orders\_prop

**Selection Predicate : acceptance\_cust\_order.**

Selection Predicate

accept\_cust\_order\_sel.\_pred.

Selected by

internal\_order\_paper7\_sact

rejected\_order\_sact

Basic Name

acceptance\_cust\_order.

Selection Predicate : delayed\_delivery.

Selection Predicate

delayed\_delivery\_sel.\_pred.

Basic Name

delayed\_delivery.

Selection Predicate : delivery.

Selection Predicate

delivery\_sel.\_pred.

Basic Name

delivery.

Selection Predicate : due/product.

Selection Predicate

due/product\_sel.\_pred.

Basic Name

due/product.

Selection Predicate : repayment.

Selection Predicate

repayment\_sel.\_pred.

Basic Name

repayment.

Selection Predicate : treatment\_cust\_order.

Selection Predicate

treatm\_cust\_order\_sel.\_pred.

Selected by

treated\_order\_sact

waiting\_orders2\_sact

Basic Name

treatment\_cust\_order.

Speech Act : accounts\_card

Speech Act

accounts\_card\_sact

Hears

information\_system\_agent

Speech Act

accounts\_card\_ia

Speech Act

accounts\_card\_prop

Speaks

accountancy\_service\_agent

Speech Act

accounts\_card\_ua

Type

virtual

Basic Name

accounts\_card

Speech Act : accounts\_card2

Speech Act

accounts\_card2\_sact

Hears

accountancy\_service\_agent

Speech Act

accounts\_card2\_ia

Speech Act

accounts\_card2\_prop

Speaks

information\_system\_agent

Speech Act

accounts\_card2\_ua

Type

virtual

Basic Name

accounts\_card2

**Speech Act : booking\_card**

Speech Act

booking\_card\_sact

Hears

central\_stock\_service\_agent

Speech Act

booking\_card\_ia

Speech Act

booking\_card\_prop

Speaks

order\_prepar\_service\_agent

Speech Act



booking\_card\_ua

Type

not virtual

Basic Name

booking\_card

### Speech Act : cheque

Speech Act

cheque\_sact

Hears

order\_recording\_service\_agent

Speech Act

cheque\_ia

Speech Act

cheque\_prop

Speaks

customer\_agent

Speech Act

cheque\_ua

Type

not virtual

Basic Name

cheque

### Speech Act : cheque2

Speech Act

cheque2\_sact

Hears

accountancy\_service\_agent

Speech Act

cheque2\_ia

Speech Act

cheque2\_prop

Speaks

order\_recording\_service\_agent

Speech Act

cheque2\_ua

Type

not virtual

Basic Name

cheque2

### Speech Act : cheque3

Speech Act

cheque3\_sact

Hears

accountancy\_service\_agent

Speech Act

cheque3\_ia

Speech Act

cheque3\_prop

Speaks

information\_system\_agent

Speech Act

cheque3\_ua

Type

virtual

Basic Name

cheque3

**Speech Act : cheque\_encashment**

Speech Act

cheque\_encashment\_sact

Hears

customer\_agent

Speech Act

cheque\_encashment\_ia

Speech Act

cheque\_encashment\_prop

Speaks

accountancy\_service\_agent

Speech Act

cheque\_encashment\_ua

Type

not virtual

Basic Name

cheque\_encashment

**Speech Act : customer\_order**

Speech Act

customer\_order\_sact

Activated by

accept/rej\_cust\_order\_stage

Hears

order\_recording\_service\_agent

Speech Act

customer\_order\_ia

Speech Act

customer\_order\_prop

Speaks

customer\_agent

Speech Act

customer\_order\_ua

Type

not virtual

Basic Name

customer\_order

Speech Act : daily\_stock\_filing\_cards

Speech Act

daily\_stock\_filing\_cards\_sact

Activated by

delivery\_stage

Hears

order\_prepar\_service\_agent

Speech Act

daily\_stock\_filing\_cards\_ia

Speech Act

daily\_stock\_filing\_cards\_prop

Speaks

information\_system\_agent

Speech Act

daily\_stock\_filing\_cards\_ua

Type

virtual

Basic Name

daily\_stock\_filing\_cards

**Speech Act : delivery\_paper**

Speech Act

delivery\_paper\_sact

Hears

central\_stock\_service\_agent

Speech Act

delivery\_paper\_ia

Speech Act

delivery\_paper\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

delivery\_paper\_ua

Type

not virtual

Basic Name

delivery\_paper

**Speech Act : delivery\_paper2**

Speech Act

delivery\_paper2\_sact

Hears

commercial\_department\_agent

Speech Act

delivery\_paper2\_ia

Speech Act

delivery\_paper2\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

delivery\_paper2\_ua

Type

not virtual

Basic Name

delivery\_paper2

Speech Act : delivery\_paper3

Speech Act

delivery\_paper3\_sact

Activated by

exhausted/sending\_stage

Hears

bill\_service\_agent

Speech Act

delivery\_paper3\_ia

Speech Act

delivery\_paper3\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

delivery\_paper3\_ua

Type

not virtual

Basic Name

delivery\_paper3

Speech Act : delivery\_paper4

Speech Act

delivery\_paper4\_sact

Hears

order\_prepar\_service\_agent

Speech Act

delivery\_paper4\_ia

Speech Act

delivery\_paper4\_prop

Speaks

bill\_service\_agent

Speech Act

delivery\_paper4\_ua

Type

not virtual

Basic Name

delivery\_paper4

Speech Act : due/product

Speech Act

due/product\_sact

Hears

order\_prepar\_service\_agent

Speech Act

due/product\_ia

Speech Act

due/product\_prop

Speaks

information\_system\_agent

Speech Act

due/product\_ua

Type

virtual

Basic Name

due/product

**Speech Act : due/product2**

Speech Act

due/product2\_sact

Hears

information\_system\_agent

Speech Act

due/product2\_ia

Speech Act

due/product2\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

due/product2\_ua

Type

virtual

Basic Name

due/product2

**Speech Act : due/product3**

Speech Act

due/product3\_sact

Hears

central\_stock\_service\_agent



Speech Act

due/product3\_ia

Speech Act

due/product3\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

due/product3\_ua

Type

not virtual

Basic Name

due/product3

**Speech Act : exhausted**

Speech Act

exhausted\_sact

Hears

customer\_agent

Speech Act

exhausted\_ia

Speech Act

exhausted\_prop

Speaks

bill\_service\_agent

Speech Act

exhausted\_ua

Type

not virtual

Basic Name

exhausted

**Speech Act : exhausted2**

Speech Act

exhausted2\_sact

Speech Act

exhausted2\_ia

Speech Act

exhausted2\_prop

Speaks

bill\_service\_agent

Speech Act

exhausted2\_ua

Type

not virtual

Basic Name

exhausted2

**Speech Act : exhausted3**

Speech Act

exhausted3\_sact

Activated by

exhausted\_note\_stage

Hears

accountancy\_service\_agent

Speech Act

exhausted3\_ia

Speech Act

exhausted3\_prop

Speaks

bill\_service\_agent

Speech Act

exhausted3\_ua

Type

not virtual

Basic Name

exhausted3

Speech Act : internal\_order\_paper

Speech Act

internal\_order\_paper\_sact

Activated by

intern\_order\_paper\_arch\_stage

Hears

order\_prepar\_service\_agent

Speech Act

internal\_order\_paper\_ia

Speech Act

internal\_order\_paper\_prop

Speaks

order\_recording\_service\_agent

Speech Act

internal\_order\_paper\_ua

Type

not virtual

Basic Name

internal\_order\_paper

Speech Act : internal\_order\_paper7

Speech Act

internal\_order\_paper7\_sact

Hears

clientele\_service\_agent

Speech Act

internal\_order\_paper7\_ia

Speech Act

internal\_order\_paper7\_prop

Speaks

order\_recording\_service\_agent

Speech Act

internal\_order\_paper7\_ua

Type

not virtual

Basic Name

internal\_order\_paper7

**Speech Act : non\_delivery\_proof**

Speech Act

non\_delivery\_proof\_sact

Hears

sending\_service\_agent

Speech Act

non\_delivery\_proof\_ia

Speech Act

non\_delivery\_proof\_prop

Speaks

bill\_service\_agent

Speech Act

non\_delivery\_proof\_ua

Type

not virtual

Basic Name

non\_delivery\_proof

### Speech Act : rejected\_order

Speech Act

rejected\_order\_sact

Hears

customer\_agent

Speech Act

rejected\_order\_ia

Speech Act

rejected\_order\_prop

Speaks

order\_recording\_service\_agent

Speech Act

rejected\_order\_ua

Type

not virtual

Basic Name

rejected\_order

### Speech Act : repayment

Speech Act

repayment\_sact

Hears

customer\_agent

Speech Act

repayment\_ia

Speech Act

repayment\_prop

Speaks

accountancy\_service\_agent

Speech Act

repayment\_ua

Type

not virtual

Basic Name

repayment

### Speech Act : sending\_documents

Speech Act

sending\_documents\_sact

Hears

sending\_service\_agent

Speech Act

sending\_documents\_ia

Speech Act

sending\_documents\_prop

Speaks

bill\_service\_agent

Speech Act

sending\_documents\_ua

Type

not virtual

Basic Name

sending\_documents

**Speech Act : sending\_note ,**

Speech Act

sending\_note\_sact

Speech Act

sending\_note\_ia

Speech Act

sending\_note\_prop

Speaks

bill\_service\_agent

Speech Act

sending\_note\_ua

Type

not virtual

Basic Name

sending\_note

**Speech Act : sending\_note2**

Speech Act

sending\_note2\_sact

Hears

sending\_service\_agent

Speech Act

sending\_note2\_ia

Speech Act

sending\_note2\_prop

Speaks

bill\_service\_agent

Speech Act

sending\_note2\_ua

Type

not virtual

Basic Name

sending\_note2

**Speech Act : sending\_note3**

Speech Act

sending\_note3\_sact

Hears

accountancy\_service\_agent

Speech Act

sending\_note3\_ia

Speech Act

sending\_note3\_prop

Speaks

bill\_service\_agent

Speech Act

sending\_note3\_ua

Type

not virtual

Basic Name

sending\_note3

**Speech Act : sending\_note4**

Speech Act

sending\_note4\_sact

Hears

information\_system\_agent



Speech Act

sending\_note4\_ia

Speech Act

sending\_note4\_prop

Speaks

accountancy\_service\_agent

Speech Act

sending\_note4\_ua

Type

virtual

Basic Name

sending\_note4

**Speech Act : sending\_note5**

Speech Act

sending\_note5\_sact

Hears

accountancy\_service\_agent

Speech Act

sending\_note5\_ia

Speech Act

sending\_note5\_prop

Speaks

information\_system\_agent

Speech Act

sending\_note5\_ua

Type

virtual

Basic Name

sending\_note5

**Speech Act : treated\_order**

Speech Act

treated\_order\_sact

Hears

information\_system\_agent

Speech Act

treated\_order\_ia

Speech Act

treated\_order\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

treated\_order\_ua

Type

virtual

Basic Name

treated\_order

**Speech Act : waiting\_orders**

Speech Act

waiting\_orders\_sact

Activated by

parcel\_constitution\_stage

Hears

order\_prepar\_service\_agent

Speech Act

waiting\_orders\_ia

Speech Act

waiting\_orders\_prop

Speaks

information\_system\_agent

Speech Act

waiting\_orders\_ua

Type

virtual

Basic Name

waiting\_orders

Speech Act : waiting\_orders2

Speech Act

waiting\_orders2\_sact

Hears

information\_system\_agent

Speech Act

waiting\_orders2\_ia

Speech Act

waiting\_orders2\_prop

Speaks

order\_prepar\_service\_agent

Speech Act

waiting\_orders2\_ua

Type

virtual

Basic Name

waiting\_orders2

Stage : accept/rej\_cust\_order\_stage

**Activates**

customer\_order

**Groups**

order\_treatment\_segm

**Suggests**

customers\_orders\_management

**Basic Name**

accept/rej\_cust\_order\_stage

**Purpose**

Acceptance or rejection of the customer order.

If the order is accepted, the internal order paper is issued and the conversation will finished by sending an "exhausted" note saying that all the products are exhausted, or by sending a parcel.

If the order is rejected, it is sending back to the customer and the conversation ends.

**Stage : delivery\_stage****Activates**

daily\_stock\_filing\_cards

**Groups**

products\_delivery\_segm

**Suggests**

customers\_orders\_management

**Basic Name**

delivery\_stage

**Purpose**

If all the products in connection with this delayed order are available or the delay is achieved, it must be delivered

**Stage : exhausted/sending\_stage****Activates**

delivery\_paper3

Groups

products\_delivery\_segm

Suggests

customers\_orders\_management

Basic Name

exhausted/sending\_stage

Purpose

Either all the order products (eventually delayed) are exhausted, or there is delivery of at least one product (even if others products of the order are exhausted)

Stage : exhausted\_note\_stage

Activates

exhausted3

Groups

customer\_delivery\_segm

Suggests

customers\_accounts\_management

Basic Name

exhausted\_note\_stage

Purpose

For each "exhausted" note received (T5), the amount to repay is taken off the accounts card if any sending note has been received for that order ; otherwise, the accountancy service grounds on the sending note from which the accounts card has been debited

Stage : intern\_order\_paper\_arch\_stage

Activates

internal\_order\_paper

Groups

order\_treatment\_segm

Suggests

customers\_orders\_management

Basic Name

intern\_order\_paper\_arch\_stage

Purpose

Archivage of the internal order paper by the preparation order service, in the waiting orders file if that order will be the object of a delayed delivery ; else it will be put into the treated orders file

Stage : parcel\_constitution\_stage

Activates

waiting\_orders

Groups

products\_delivery\_segm

Suggests

customers\_orders\_management

Basic Name

parcel\_constitution\_stage

Purpose

If the delayed order can't be totally fulfilled, the due by product card is sent to the stock central service

Topicalization : booked\_quantities..

Concerned by

booking\_card

Basic Name

booked\_quantities..

Topicalization : cheque/payment..

## Characterized by

cheque\_treatment\_segm

## Concerned by

accounts\_card  
accounts\_card2  
cheque  
cheque2  
cheque3  
cheque\_encashment

## Basic Name

cheque/payment..

Topicalization : customer\_delivery..

## Characterized by

customer\_delivery\_segm  
products\_delivery\_segm

## Concerned by

delivery\_paper  
delivery\_paper2  
delivery\_paper3  
delivery\_paper4  
non\_delivery\_proof  
sending\_documents  
sending\_note  
sending\_note2  
sending\_note3  
sending\_note4  
sending\_note5

## Basic Name

customer\_delivery..

Topicalization : customer\_order..

## Characterized by

order\_treatment\_segm

## Concerned by

customer\_order  
internal\_order\_paper

internal\_order\_paper7  
rejected\_order  
treated\_order  
waiting\_orders  
waiting\_orders2

Basic Name

customer\_order..

Topicalization : due\_quantities..

Concerned by

due/product  
due/product2  
due/product3

Basic Name

due\_quantities..

Topicalization : repayment..

Concerned by

exhausted  
exhausted2  
exhausted3  
repayment

Basic Name

repayment..

Topicalization : stock..

Concerned by

daily\_stock\_filing\_cards

Basic Name

stock..

Triggering Predicate : cheque.

Triggering Predicate



cheque\_trig.\_pred.

Triggered by

cheque\_convers

Basic Name

cheque.

Triggering Predicate : customer\_order.

Triggering Predicate

customer\_order\_trig.\_pred.

Triggered by

customer\_order\_convers

Basic Name

customer\_order.

Triggering Predicate : exhausted\_product.

Triggering Predicate

exhausted\_product\_trig.\_pred

Triggered by

exhausted\_product\_convers

Basic Name

exhausted\_product.

Triggering Predicate : sending\_note.

Triggering Predicate

sending\_note\_trig.\_pred.

Basic Name

sending\_note.

**Triggering Predicate : stock\_cards.**

Triggering Predicate

stock\_cards\_trig.\_pred.

Triggered by

stock\_cards\_convers

Basic Name

stock\_cards.

**Utterance Act : accounts\_card2\_ua**

Represents Utterance Act

accounts\_card2\_prop

Utterance Act

accounts\_card2

Basic Name

accounts\_card2\_ua

Form

written

**Utterance Act : accounts\_card\_ua**

Represents Utterance Act

accounts\_card\_prop

Utterance Act

accounts\_card

Basic Name

accounts\_card\_ua

Form

written

**Utterance Act : booking\_card\_ua**

Represents Utterance Act

booking\_card\_prop

Utterance Act

booking\_card

Basic Name

booking\_card\_ua

Form

written

Utterance Act : cheque2\_ua

Represents Utterance Act

cheque2\_prop

Utterance Act

cheque2

Basic Name

cheque2\_ua

Form

written

Utterance Act : cheque3\_ua

Represents Utterance Act

cheque3\_prop

Utterance Act

cheque3

Basic Name

cheque3\_ua

Form

written

**Utterance Act : cheque\_encashment\_ua**

Represents Utterance Act

cheque\_encashment\_prop

Utterance Act

cheque\_encashment

Basic Name

cheque\_encashment\_ua

Form

manual

**Utterance Act : cheque\_ua**

Represents Utterance Act

cheque\_prop

Utterance Act

cheque

Basic Name

cheque\_ua

Form

written

**Utterance Act : customer\_order\_ua**

Represents Utterance Act

customer\_order\_prop

Utterance Act

customer\_order

Basic Name

customer\_order\_ua

Form

written

**Utterance Act : daily\_stock\_filing\_cards\_ua**

Represents Utterance Act

daily\_stock\_filing\_cards\_prop

Utterance Act

daily\_stock\_filing\_cards

Basic Name

daily\_stock\_filing\_cards\_ua

Form

written

**Utterance Act : delivery\_paper2\_ua**

Represents Utterance Act

delivery\_paper2\_prop

Utterance Act

delivery\_paper2

Basic Name

delivery\_paper2\_ua

Form

written

**Utterance Act : delivery\_paper3\_ua**

Represents Utterance Act

delivery\_paper3\_prop

Utterance Act

delivery\_paper3

Basic Name

delivery\_paper3\_ua

Form

written

Utterance Act : delivery\_paper4\_ua

Represents Utterance Act

delivery\_paper4\_prop

Utterance Act

delivery\_paper4

Basic Name

delivery\_paper4\_ua

Form

written

Utterance Act : delivery\_paper\_ua

Represents Utterance Act

delivery\_paper\_prop

Utterance Act

delivery\_paper

Basic Name

delivery\_paper\_ua

Form

written

Utterance Act : due/product2\_ua

Represents Utterance Act

due/product2\_prop

Utterance Act

due/product2

Basic Name

due/product2\_ua

Form

written

Utterance Act : due/product3\_ua

Represents Utterance Act

due/product3\_prop

Utterance Act

due/product3

Basic Name

due/product3\_ua

Form

written

Utterance Act : due/product\_ua

Represents Utterance Act

due/product\_prop

Utterance Act

due/product

Basic Name

due/product\_ua

Form

written

Utterance Act : exhausted2\_ua

Represents Utterance Act

exhausted2\_prop

Utterance Act

exhausted2

Basic Name

exhausted2\_ua

Form

written

**Utterance Act : exhausted3\_ua**

Represents Utterance Act

exhausted3\_prop

Utterance Act

exhausted3

Basic Name

exhausted3\_ua

Form

written

**Utterance Act : exhausted\_ua**

Represents Utterance Act

exhausted\_prop

Utterance Act

exhausted

Basic Name

exhausted\_ua

Form

written

**Utterance Act : internal\_order\_paper7\_ua**

Represents Utterance Act



internal\_order\_paper7\_prop

Utterance Act

internal\_order\_paper7

Basic Name

internal\_order\_paper7\_ua

Form

written

Utterance Act : internal\_order\_paper\_ua

Represents Utterance Act

internal\_order\_paper\_prop

Utterance Act

internal\_order\_paper

Basic Name

internal\_order\_paper\_ua

Form

written

Utterance Act : non\_delivery\_proof\_ua

Represents Utterance Act

non\_delivery\_proof\_prop

Utterance Act

non\_delivery\_proof

Basic Name

non\_delivery\_proof\_ua

Form

written

Utterance Act : rejected\_order\_ua

Represents Utterance Act

rejected\_order\_prop

Utterance Act

rejected\_order

Basic Name

rejected\_order\_ua

Form

written

#### Utterance Act : repayment\_ua

Represents Utterance Act

repayment\_prop

Utterance Act

repayment

Basic Name

repayment\_ua

Form

manual

#### Utterance Act : sending\_documents\_ua

Represents Utterance Act

sending\_documents\_prop

Utterance Act

sending\_documents

Basic Name

sending\_documents\_ua

Form

written

**Utterance Act : sending\_note2\_ua**

Represents Utterance Act

sending\_note2\_prop

Utterance Act

sending\_note2

Basic Name

sending\_note2\_ua

Form

written

**Utterance Act : sending\_note3\_ua**

Represents Utterance Act

sending\_note3\_prop

Utterance Act

sending\_note3

Basic Name

sending\_note3\_ua

Form

written

**Utterance Act : sending\_note4\_ua**

Represents Utterance Act

sending\_note4\_prop

Utterance Act

sending\_note4

Basic Name

sending\_note4\_ua

Form



written

**Utterance Act : sending\_note5\_ua**

Represents Utterance Act

sending\_note5\_prop

Utterance Act

sending\_note5

Basic Name

sending\_note5\_ua

Form

written

**Utterance Act : sending\_note\_ua**

Represents Utterance Act

sending\_note\_prop

Utterance Act

sending\_note

Basic Name

sending\_note\_ua

Form

written

**Utterance Act : treated\_order\_ua**

Represents Utterance Act

treated\_order\_prop

Utterance Act

treated\_order

Basic Name

treated\_order\_ua

Form

written

Utterance Act : waiting\_orders2\_ua

Represents Utterance Act

waiting\_orders2\_prop

Utterance Act

waiting\_orders2

Basic Name

waiting\_orders2\_ua

Form

written

Utterance Act : waiting\_orders\_ua

Represents Utterance Act

waiting\_orders\_prop

Utterance Act

waiting\_orders

Basic Name

waiting\_orders\_ua

Form

written