الجمهورية الجزائرية الديموقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab de Blida

FACULTE DE SCIENCES EXACTES

DEPARTEMENT D'INFORMATIQUE

Mémoire de projet de fin d'études
En vu d'obtention du diplôme
D'ingénieur d'état en informatique

Option : base de données avancée

Présenté par :

M^lle BOUMAHDI Fatima

M^lle FAREH Messaouda

Thème

ANALYSE ET CONCEPTION D'UN OUTIL DE DEFINITION DE WORKFLOW BASE SUR LE LANGAGE EXPRESS

Soutenu le 15 juillet devant le jury composé de :

M^me.OUKID	Présidente
M^me.ABED	Promotrice
M^r HAMDAH Mohamed	Encadreur
M^me.BENSTITI	Examinatrice
M^r BEN NOUAR	Examinadeur

Année universitaire 2002-2003

Remerciements

Nous remercions avant tout le bon dieu qui nous a aidé à réaliser ce modeste travail.

Nous tenons à remercier notre promoteur M'. HAMDAH Mohamed pour son aide, sa patience, sa disponibilité et sa compréhensibilité.

Nous remercions les membres du jury pour nous avoir fait l'honneur de juger notre travail.

Nous adressons nos sincères remerciements à Mme.OUAHRANI et Mme.ABED pour leurs disponibilité, pour leurs aides ainsi que pour leurs conseils durant notre projet.

Nous tenons à exprimer nos profondes gratitudes et nos vifs remerciements à M'.BELKAID Ismail, M'.El Hadj MIMOUNE Mourad et M'.DEHAINSALA Hondjack pour leurs aides, leurs conseils et leurs préoccupation, durant notre projet, ainsi leurs aides morale.

Nous remercions aussi M'.BOUMAHDI Mohamed, M'.BOUMAHDI Abdel karim, et M'.ZEBBICHE Djamel, pour leurs aides et leurs encouragements le long du projet.

Nous remercions tous les enseignants de la faculté des sciences de BLIDA et surtout ceux du département informatique.

Nous remercions, de tout coeur, tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Dédicace

Ce mémoire est dédié :

A mes très chers ***parents*** *pour leur soutien durant toute ma carrière,*

Pour leurs bienveillances, leurs efforts constants dans mes études, et Pour leurs encouragements,

A mon grand frère ***Mohamed*** *pour m'avoir soutenu durant mes études, et son fils : mon très cher et adorable le petit prince* ***Hichem Abdel Hafidh.***

A mes frères ***Mostapha, Abdel Karim, Salem*** *et* ***Omar.***

A mes seours ***Wahiba, Nassima et Nessrine.***

A tous mes nièces, et en particulier la vedette ***Rim***

A toute la famille ***BOUMAHDI,***

A mon binôme ***Messaouda*** *et à toute sa famille*

A ***Ismail, Fatiha*** *et* ***Drai Akila.***

A tous mes amis.

BOUMAHDI Fatima (Djauida)

Dédicace

Ce mémoire est dédié :

A mes très chers parents pour leurs soutiensdurant

toute ma carrière,

Pour leurs bienveillances, leurs efforts constants dans mes études,

Et Pour leurs encouragements,

A mon frère Youcef,

A ma soeur Nora,

A toute ma famille,

A mon binôme Fatima et à toute sa famille

A Amina, Nadia, Assia, Habiba, Nabila, Aicha , Wafaa,

Ghania et Ismail.

A tous mes amis.

F.Messaouda

Analyse et conception d'un outil de définition de workflow basé sur le langage EXPRESS.

Résumé. L'arrivée des nouveaux outils de communication a entraîné de nouvelles technologies, elles comprennent le groupware, dans cette technologie, le workflow fait partie des environnements les plus puissants pour automatiser les processus de travail. Le workflow est un mode de division du travail entre les rôles, qui facilite la coordination des actions, pour cela, il faut définir un système qui gère et contrôle les différentes activités, c'est le système de gestion de workflow.

Notre travail est intéressé par la première phase de construction d'un système de gestion de workflow : c'est l'analyse de workflow. Durant cette phase il faut développer un outil qui permet la définition de workflow.

Au cours de notre projet nous avons développé un outil de définition, simple et convivial d'utilisation permettant aux usagers de représenter graphiquement leur métier, leurs activités, leurs enchaînements et les règles associées, dans un formalisme dépend du méta modèle correspondant au type d'activité modélisé.

Mots clés. Groupware, Workflow, EXPRESS, UML (unified modeling language), modélisation orienté objet , Ecco Toolkit.

Analysis and design of a workflow definition tool using the EXPRESS language.

Abstract. The arrival of the new tools of communication dragged new technologies. Among these technologies, we were interested by the groupware and especially by the workflow. The workflow makes the most powerful environment part to automate processes of work. The workflow permits the division of work between the different roles. This division facilitates the actors coordination. That is why, it is necessary to define a system of workflow management.

Our work constitutes the first construction's phase of a system of workflow management: it is the analysis of workflow. During this phase it is necessary to develop a tool that permits the definition of workflow.

During our project we developed a tool of definition, simple and convivial, allowing users to represent their profession, their activities, their sequences and the associated rules, in a formalism closed by the meta models corresponding to the type of a modal activity.

Key words. Groupware, workflow, EXPRESS, UML (unified modeling language), modeling oriented object, Ecco Toolkit.

TABLE DES MATIERES

LISTE DES TABLEAUX

CHAPITRE I

CHAPITRE II

CHAPITRE III

ANNEXE B

LISTE DES FIGURES

CHAPITRE I

CHAPITRE II

CHAPITRE III

Analyse et conception d'un outil de définition de workflow basé sur le langage EXPRESS.

Résumé. L'arrivée des nouveaux outils de communication a entraîné de nouvelles technologies, elles comprennent le groupware, dans cette technologie, le workflow fait partie des environnements les plus puissants pour automatiser les processus de travail. Le workflow est un mode de division du travail entre les rôles, qui facilite la coordination des actions, pour cela, il faut définir un système qui gère et contrôle les différentes activités, c'est le système de gestion de workflow.

Notre travail est intéressé par la première phase de construction d'un système de gestion de workflow : c'est l'analyse de workflow. Durant cette phase il faut développer un outil qui permet la définition de workflow.

Au cours de notre projet nous avons développé un outil de définition, simple et convivial d'utilisation permettant aux usagers de représenter graphiquement leur métier, leurs activités, leurs enchaînements et les règles associées, dans un formalisme dépend du méta modèle correspondant au type d'activité modélisé.

Mots clés. Groupware, Workflow, EXPRESS, UML (unified modeling language), modélisation orienté objet , Ecco Toolkit.

Analysis and design of a workflow definition tool using the EXPRESS language.

Abstract. The arrival of the new tools of communication dragged new technologies. Among these technologies, we were interested by the groupware and especially by the workflow. The workflow makes the most powerful environment part to automate processes of work. The workflow permits the division of work between the different roles. This division facilitates the actors coordination. That is why, it is necessary to define a system of workflow management.

Our work constitutes the first construction's phase of a system of workflow management: it is the analysis of workflow. During this phase it is necessary to develop a tool that permits the definition of workflow.

During our project we developed a tool of definition, simple and convivial, allowing users to represent their profession, their activities, their sequences and the associated rules, in a formalism closed by the meta models corresponding to the type of a modal activity.

Key words. Groupware, workflow, EXPRESS, UML (unified modeling language), modeling oriented object, Ecco Toolkit.

Introduction générale

Le temps change, le monde aussi, ce monde a connu, en particulier ce dernier siècle, des révolutions technologiques qui ne cessent de changer les méthodes de travail des organisations, surtout devant les évolutions de l'environnement technologique et économique, elles doivent être de plus en plus réactives pour obtenir la satisfaction du client qui se trouve maintenant placée au centre de leurs préoccupations. Si une organisation en processus est considérée comme une solution très prometteuse, il s'avère pourtant que sa mise en œuvre n'a abouti le plus souvent qu'à une simple automatisation des processus.

La technologie du groupware recouvre les domaines vastes de coopération, communication, et de collaboration. Elle est la solution la plus adaptée pour informatiser les processus métier. La technologie du workflow est centrée essentiellement sur les processus métier des entreprises et permet l'automatisation de processus de travail.

Nous nous arrêterons sur la technologie workflow, comprise comme une application groupware, qui regroupe toute la problématique de l'information et de la communication.

Le travail que nous allons présenter rentre dans un projet lancé par le Centre de Recherche sur l'Information Scientifique et Technique (CERIST), et qui consiste à développer un système de gestion de workflow à base du langage EXPRESS. Dans ce cadre y est intégré notre projet, notre rôle est d'être responsable de la première phase par la quelle doit passer le développement du SGWF[1], c'est la phase d'analyse.

Notre travail consiste à développer un outil de définition de workflow, afin de permettre une représentation graphique des processus métier par des non informaticiens (des experts des métiers), ainsi que des définitions associées à ces processus.

Afin d'atteindre le principal but de notre étude, nous avons organisé notre mémoire comme suit :

Le premier chapitre sera consacré à l'étude d'un nouveau mode de fonctionnement des entreprises plus flexibles basé sur le travail coopératif soit le groupware et d'un type d'application : le workflow. Nous présenterons pour ce dernier ses concepts fondamentaux ainsi que ses différents types d'applications.

[1] **SGWF** : Système de Gestion de WorkFlow

Dans le deuxième chapitre, une brève présentation de l'approche orienté objet et les concepts fondamentaux de la modélisation seront présentés, ainsi qu'une étude de deux langages reconnus dans la modélisation et la spécification de données, UML(unified modeling language) et EXPRESS. Ces derniers seront utilisés dans le développement de l'outil.

Après avoir posé quelques repères théoriques sur ces nouvelles technologies, nous aborderons dans le troisième chapitre le développement de l'outil, suivant le cycle de vie en cascade, en utilisant le langage de modélisation UML. Nous verrons dans un premier temps, les fonctionnalités que le système doit offrir aux usagés. Puis, dans un deuxième temps, une conception de l'outil sera bien détaillée. Enfin, nous passerons à l'implémentation de la solution retenue au niveau de la conception.

Nous achèverons notre mémoire par une conclusion générale, présentant l'importance de notre outil dans un système de gestion de workflow et dégageant les perspectives envisagées à cours et à moyen terme.

1. La problématique

Ce présent travail consiste à résoudre un certain nombre de problèmes, qui sont résumés en :

- La difficulté de l'automatisation des processus. Cette difficulté emmène les entreprises à intégrer le workflow dans leurs procédures de travail, et nécessite donc des outils de workflow.
- La difficulté de la modélisation des processus par rapport aux experts du métier dans l'entreprise, sachant que leurs connaissances en matière informatique sont limitées,
- Les problèmes d'interprétation permettent le passage à des structures implémentables. Ces problèmes sont dûs aux faits que l'ensemble des notations est semi formel, ce qui constitue un handicap pour leur traitement automatique.

2. Les objectifs

Face à ces différents problèmes, nous avons mis en œuvre notre outil, dont l'objectif est d'apporter des solutions pertinentes aux problèmes énoncés.

Notre objectif est de développer un outil de définition de workflow, simple et convivial, permettant une utilisation facile. Cet outil offre les fonctionnalités suivantes :

- Définition de nouveaux processus et mise à jour des processus existants à travers la modélisation, la consultation et la modification d'un processus de workflow.
- Extensibilité : possibilité avec laquelle un modèle de processus pourra être adapté pour faire face à une évolution des besoins de l'utilisateur.
- Réutilisation des modèles: possibilités d'utiliser même définition de processus dans plusieurs applications.
- Utilisation d'une notation graphique, pour permettre à tous les travailleurs de l'organisation de participer à la construction d'une vue commune et claire des processus.
- Rendre les modèles de workflow générés par notre outil exécutables par des moteurs de workflow, en nous appuyant sur la génération des représentations textuelles interprétables.
- Garantie de la désintermédiation pour permettre aux experts métier d'analyser et de modéliser les processus de l'entreprise sans l'intervention d'un informaticien.

CHAPITRE I. GROUPWARE ET WORKFLOW

1. Introduction

Les évolutions technologiques ont un impact certain sur l'entreprise, et ont entraîné celle-ci à modifier son organisation et à reconfigurer ses processus pour être plus rentable. De cela, est née une nouvelle génération de SI[2], qui intègre la composante « coopération ». Ces systèmes font appel à la technologie du *groupware* de façon générale, et à celle du *workflow* en particulier.

Afin de définir précisément ces technologies nous avons organisé ce chapitre comme suit: la première section consiste à présenter la technologie du groupware, la seconde est consacrée à la description de l'application de workflow, en présentant également leur impacts sur les entreprises.

2. Le groupware

La réflexion sur le groupware est née aux Etats-Unis en 1968 à la suite des recherches universitaires sur le travail coopératif assisté par ordinateur [Saadoun, 96]. Il s'appelait alors le CSCW[3] .

Le terme 'groupware' est apparu pour la première fois dans un article de la revue Fortune "Software catches than spirit" en 1987 [khochafiane &al, 98]. L'année 1992 voit l'organisation de la première grande conférence sur le groupware [khochafiane &al, 98].

Selon Peter et Trudy Johnson Lez[4] (1978) une définition du groupware est la suivante : « est un processus de travail en groupe visant un objectif précis et logiciels conçu, pour faciliter ce travail en groupe» [Saadoun, 96].

L'entreprise doit tendre à devenir un espace de communication, de collaboration et de coordination [Blau, 01]. Il favorise les changements de mode de travail et de groupe de travail dans le temps.

Pour réussir, le projet groupware touche à trois dimensions de l'entreprise, le management ou la dimension humaine, l'organisation ou la dimension organisationnelle, l'informatique ou la dimension technologique.

---

[2] **SI** : Système d'Information
[3] **CSCW**:Computer Supported Cooperativ Work
[4] Peter et Trudy Johnson-Lenz : sont deux chercheurs du New Jersey Institute of Technology.

2.1. Les services

Le groupware est un concept qui porte avant tout sur les processus de communication et de travail en groupe, et sur la façon dont ces processus peuvent être soutenus par des outils logiciels basés sur une architecture réseau: réseau, puisque le but est d'interconnecter plusieurs personnes entre elles. Le groupware couvre les fonctions des trois C :

•**Communication :** par la distribution et le partage des informations.

•**Collaboration:** par un travail commun, l'échange d'idées et le partage de connaissances.

•**Coordination :** par la synchronisation des tâches.

2.1.1. La communication

Les besoins de communication et d'échange sont fondamentaux. Ces échanges entre membres d'une équipe peuvent être plus ou moins structurés selon que les besoins de coordination ou de coopération dominent. On peut réaliser la communication par : la messagerie électronique, les forums électroniques…

a) La messagerie électronique

La communication qui se fait par la messagerie, forme le cœur de tout système groupware. Les messages sont conservés et forment une base de données permanente de la connaissance de l'entreprise, son carnet d'adresses est utilisé pour organiser des réunions de groupe, router des documents et accorder des droits d'accès aux documents.

La messagerie permet maintenant d'envoyer non plus le document, mais un lien vers lui, améliorant ainsi le partage des documents.

b) Les forums électroniques

Ceux-ci permettent de communiquer et de partager des connaissances. En offrant une vue commune aux participants. Ces outils permettent de leur donner une même base d'informations au début de leur travail en commun. De plus, ils vont faciliter les prises de décisions en groupe et la création d'idées nouvelles.

2.1.2. La collaboration

Dans les bases de données partagées on utilise un Système "pull"[5]. Le problème qui se pose est : comment savoir qu'une information a été modifié ou non? Donc il faut posséder la dernière mise à jour de cette information, car cette dernière vient d'être reliée à de nouveaux documents. C'est le but de la messagerie qui, intégrée à ces bases de données, pourra informer les utilisateurs de tout changement.

2.1.3. La coordination

Afin de router les documents entre plusieurs personnes, un workflow simple, bien que reliant les membres d'un groupe entre eux, va les laisser isolés en les privant d'une vue d'ensemble du processus, chacun étant cantonné à sa tâche.

Y inclure la messagerie pour notifier tout changement ainsi qu'une base de données partagée pour stocker les éléments avec leurs états d'avancement, va permettre à quiconque ayant droit d'accès, de se renseigner sur le contexte. Donc le workflow automatise des procédures formelles de travail et améliore la communication de l'entreprise [khochafiane &al, 98].

[5] **Pull**:un système où chacun va y chercher l'information dont il a besoin au moment désiré.

2.2. Les règles d'or du groupware

Le groupware touche tous les domaines constituant l'entreprise. A savoir les hommes et leur management, l'organisation et les processus de travail et les outils informatiques.

Nous pouvons mentionner trois règles de base [Saadoun, 96] qui aideront à situer un système groupware dans ce contexte, et qui rappelleront son importance et son impact dans l'entreprise.

2.2.1. Etre conscient que le groupware n'est pas qu'un produit logiciel

Le groupware est un cocktail composé du 1/3 Management, du 1/3 Organisation et du 1/3 Informatique [Levan, 99a]. Cette métaphore illustre la nécessité prise en compte de la dimension humaine, de la dimension organisationnelle et de la dimension informatique dans le but est l'introduction et le développement des technologies du travail collaboratif en entreprise.

- **Changement sur le management**

Parce qu'il faudra apprendre aux employés à travailler en équipe et à accepter de partager leurs connaissances et leurs informations avec d'autres. Si les individus ne le font pas, le groupware ne sert à rien, et pourra même être contre-productif.

- **Changement sur l'organisation**

L'entreprise peut être amenée à devoir reconfigurer ses processus afin de les adapter au travail en groupe. Ceci dans le but d'améliorer ses processus, afin de mieux réagir à son environnement et de répondre mieux et plus vite aux clients.

- **Changement sur l'informatique**

L'architecture du groupware doit s'appuyer sur le système d'information de l'entreprise. Cependant, si les systèmes de communication ne sont pas adaptés pour supporter un système groupware, il faudra penser à changer l'architecture réseau de l'entreprise [Saadoun, 96]. De plus, il se peut que l'entreprise doive équiper certains postes de travail, en matériel informatique, et former les utilisateurs.

2.2.2. Un projet groupware est un véritable projet de management

Il faut considérer, dés le départ l'introduction du groupware comme un projet de management, par ce que le groupware n'est pas qu'une affaire d'informaticiens, loin de là.

L'introduction du groupware devrait être le résultat d'une réflexion stratégique elle même en fonction des objectifs à attendre au niveau de l'entreprise.

Si le groupware doit être le résultat d'une réflexion de management stratégique, il doit également servir les objectifs du management opérationnel et apporter les résultats attendus relativement à la performance des processus critiques de l'entreprise. Pour ce la il faut respecter la première règle.

2.2.3. Un projet groupware est un véritable processus de changement

Ce troisième point est la conséquence de tout ce qui a été vu jusqu'ici. Cela demande de travailler avec méthode, tout en restant flexible, et de respecter les trois temps caractérisant les processus de changement qui sont:

- Réfléchir sur la cible et définir la vision de l'objectif qui seule provoque et justifie le changement : il faut d'abord bien identifier le futur, c'est-à-dire la situation cible sur les plans humains, organisationnels, et technologiques.

•Identifier les gains, les points forts et ces qu'il y a à améliorer, afin d'atteindre la cible définie : il faut ensuite évaluer le présent, avec les points forts et les axes d'amélioration, toujours sur les plans humains, organisationnels, et technologiques.

•Préparer la transition: il faut finalement organiser la transition, en veillant à raccourcir les étapes quitte à les multiplier [Levan, 99a], et surtout à réunir toutes les conditions pour que chaque efforts du changement soit un succès.

2.3. Typologie des applications groupware

Etablir la typologie des applications groupware va servir à montrer la différence et la variété des catégories dans lesquelles se classent les outils du groupware, mais aussi et surtout va faire ressortir leur complémentarité [Frey, 99]. Ceci aidera au choix du meilleur système groupware pour l'entreprise. Il est nécessaire de comprendre et de différencier les technologies du groupware avant de les choisir. En effet, certains produits sont mieux adaptés que d'autres aux objectifs visés par l'entreprise.

Pour répondre aux objectifs de communication, de coopération et de coordination, cette classification nous offre le routage, la mémoire, et l'échange (Tableau I-1). Le degré de structure de l'entreprise et le niveau de formalisme de l'interactivité distingue les catégories groupware :

- La mémoire organisationnelle qu'il faut maintenir et expliquer.
- Le routage pour la diffusion des documents.
- L'échange pour la consultation des documents.

	Applications de groupware orientées Mémoire	**Applications de groupware orientées Routage**	**Applications de groupware orientées Echange**
+ Structuré	**Bibliothèque** Bibliothèque de projet Bibliothèque de groupe Bibliothèque de formulaire	**Workflow** Tous processus administratif et de production (exemple: instruction d'un dossier de prêt bancaire)	**Suivi** Calendrier de groupe Gestion et suivi de tâches Suivi de contrat
- Structuré	**Kiosque** Journal d'entreprise Annonces de postes vacants Revue de presse	**Messagerie** Courrier électronique Formulaire électronique	**Conférence** Réunion de coordination Préparation/suite de réunion

Tableau I-1: Classification des outils groupware [Frey, 99]

2.3.1. La famille des applications orientées mémoire

Cette famille contient toutes les applications groupware dont le but principal est de mettre en commun des informations et des connaissances, qui constituent la mémoire collective du groupe dont les supports sont des documents, qu'il s'agisse de textes, d'images fixes ou animées ou de sons.

Dans cette typologie des applications groupware, nous distinguons deux sous-classes d'applications héritées de la classe générique :

1) Les applications de bibliothèques

Cette sous-classe d'application repose sur les technologies utilisées par les systèmes de gestion

documentaires. Les bases de document partagées permettent de suivre la transformation historique de chaque document depuis sa création jusqu'à sa dernière version.

2) Les applications de Kiosque

Les applications de type Kiosque constituent des sortes de présentoirs de journaux électroniques dont lesquels les membres du groupe savent retrouver des événements particuliers, ponctuels mais concernant l'univers d'activité du groupe. La différence avec l'application de bibliothèque réside dans la nature de l'information qui transite dans le kiosque : articles, communiqués de presse, lettres d'information…etc.

2.3.2. La famille des applications orientées routage

Cette famille d'application va permettre à chaque participant et au groupe de développer des modes particulièrement efficaces de communication et de coordination. La transmission des informations s'effectue ici sur un mode asynchrone, c'est à dire qu'elle n'exige pas qu'il y ait unité de temps pour que la communication s'établisse. Elle fonctionne sur ce qu'on, appelle le mode "push"[6].

Toutefois, le mode push atteint rapidement ces limites lorsque les interactions entre les personnes deviennent plus nombreuses.

Nous distinguons deux sous-classes d'application de routage :

1) Les applications de workflow

Les applications de workflow permettent à un groupe d'automatiser des processus de travail structurés et prédéterminés [Blau, 01]. Pour l'accomplissement d'une tâche ou visant à ce que qu'un travail soit effectué de manière précise.

Une application de workflow est un système très actif qui fait circuler les informations entre les personnes et qui spécifie des tâches à accomplir [Frey, 99].

Le workflow ayant, l'automatisation et le pilotage des activités au sein d'un processus, les applications de workflow jouent un rôle fondamental dans la reconfiguration et la redéfinition des processus dans l'entreprise.

2) Les applications de messagerie

La messagerie est une technologie ancienne née au moment où l'on a su relier deux ordinateurs entre eux. Si depuis les interfaces et les techniques de messagerie ont beaucoup changé, le principe de fonctionnement est lui resté le même.

Une application de messagerie reçoit et stocke des messages, les transporte et les délivre aux destinataires lorsqu'ils sont prêts à les recevoir. Ce principe à révolutionner les modes de communication car son principal avantage réside dans son mode de connexion asynchrone [Frey,99].

Les messages comportent du texte, des images, des sons mais aussi des pointeurs vers des ressources partagées (en fait des bases de documents partagés) sur les postes de travail ou sur les serveurs.

Les applications de messagerie vont du simple courrier électronique aux applications groupware ou workflow les plus complexes [Saadoun, 02]. En effet, le module de transport de la messagerie prend en charge tous les déplacements de message. Cela signifie que toutes les applications groupware exploitent des services du transport de messages, des carnets d'adresses

[6] **Mode push**:c'est l'émetteur qui décide de la transmission de l'information vers un ou plusieurs destinataires.

(pour localiser les personnes) et des bases d'objets (messages ou autres documents).

2.3.3. La famille des applications orientées "Echange"

Toutes les applications groupware dont le but principal est de supporter de façon totalement asynchrone, les interactions entre plusieurs acteurs impliqués dans des actions communes, et quel que soit le lieu ou le moment des ces interactions.

La classe générique des applications orientées échange est celle qui justifie le plus la raison d'être des applications groupware [Levan, 99a]. En effet, les processus de travail en groupe sont dominés par des échanges intenses entre les personnes. Dans les environnements de travail complexes, où la division du travail fait largement appel aux mécanismes de coordination, les besoins de communication et d'échange sont fondamentaux. Ces échanges entre membres d'une équipe sont plus ou moins structurés selon que les besoins de coordination ou de coopération exigent.

Il y a deux sous-classes d'applications d'échange :

- **Les applications de suivi**

Les applications groupware de cette catégorie ont pour but d'assurer aux membres d'une équipe une communication et un suivi permanent relatifs aux activités quotidiennes. Ces mécanismes de coordination font appel à quatre processus principaux :

- Détermination des objectifs dans le cadre d'une mission donnée.
- Décomposition des objectifs en tâches et en étapes.
- Désignation des personnes et affectation des tâches aux personnes.
- Gestion des interdépendances entre activités et personnes.

Les applications de suivi ont pour but de faciliter ces types d'échange essentiellement concentrés sur la tâche à accomplir au sein de l'équipe. Elles servent à synchroniser les activités, rendez-vous, contacts, réunions, travaux coopératifs…etc.

Voici un exemple d'application de suivi dans un environnement groupware :

- **L'agenda de groupe**, qui permet à plusieurs personnes de suivre à la fois leurs propres activités personnelles et quotidiennes, tout en partageant leurs plannings avec les autres membres du groupe.

Le suivi permet à plusieurs personnes:

➢ D'enregistrer et de suivre des documents et des informations relatifs aux produits et services.

➢ De coordonner leurs plannings.

➢ De partager des ressources collectives et de suivre des affaires, facilitant ainsi la coordination et la communication entre les personnes et les équipes.

- **Les applications de conférence**

Il s'agit des applications qui favorisent le plus la coopération entre les membres d'un groupe ou d'une équipe. Les conférences se caractérisent par leurs capacités à organiser des interactions entre plusieurs personnes à la fois selon des critères de temps et de lieu.

Il y a plusieurs raisons pour lesquelles les membres d'un groupe peuvent être amenés à se réunir : pour partager des informations, pour prendre des décisions.

Les applications de conférences électroniques ont donc créé un changement radical dans la

manière dont ce fait la communication quotidienne. La façon dont on conduit une réunion change, ainsi que la façon dont on travaille dans cette réunion de groupe [Frey, 99].

La figure suivante résume les différents types d'applications de groupware.

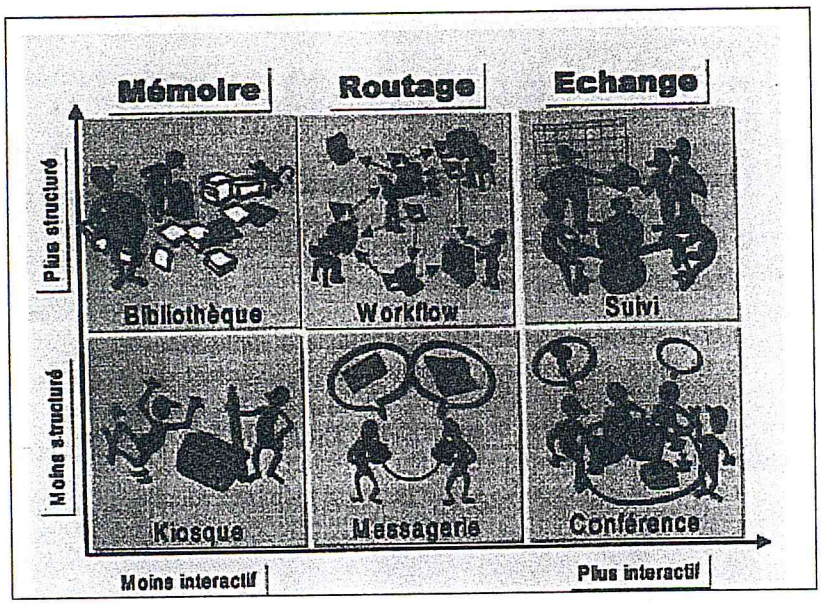


Figure I-1: La typologie des applications groupware [Melissa, 96]

3. Le workflow

Dans les technologies du groupware, les logiciels de workflow font partie des environnements les plus puissants pour automatiser les processus de travail. En effet, l'utilisation d'un système workflow dans l'entreprise va permettre la modélisation des procédures de travail, et de modifier son organisation et de reconfigurer ses processus pour les orienter vers les clients.

Le travail réellement accompli par les usagers du workflow ne consiste pas uniquement en un échange formel et procédural. Les échanges informels sont toujours présents et prennent toute leur importance lorsque la complexité des tâches et des processus demande l'intelligence collective du groupe [Saadoun, 96]. Si l'on reprend la terminologie du groupware, nous remarquons que le workflow est tout d'abord une pièce de cette technologie, servant globalement les usages de communications variées inhérentes aux situations du travail coopératif, mais qu'en plus, cette pièce est centrale.

La réalité exige l'accompagnement du workflow par d'autres applications groupware, soit:

- Une bibliothèque pour enrichir le workflow d'une mémoire organisationnelle, dans laquelle les individus pourront chercher des connaissances nécessaires à leur prise de décision.
- De conférences électroniques, afin d'équiper le workflow d'outils spécifiques des modes de communication informelle, lieu d'échange et de discussion, permettant de traiter les points du projet qui posent problème.

3.1. Définition

La WfMC[7] défini le workflow comme l'automatisation de tout ou partie d'un processus d'entreprise au cours duquel l'information circule d'une activité à l'autre, c'est-à-dire d'un participant (ou d'un groupe de participants) à l'autre pour action, en fonction d'un ensemble de

[7] **WfMC** : Workflow Management Coalision

règles de gestion" [Wfmc, 95] .

Quelque soit la définition de workflow, tout le monde s'accorde sur un point : la technologie workflow améliore la productivité. Par productivité, on entendra un service de meilleure qualité et plus rapide.

3.2. Notion de processus

Selon la WfMC, le processus d'entreprise est « un ensemble de plusieurs activités reliées les unes aux autres pour réaliser un objectif généralement, dans un contexte organisationnel qui définit des rôles et des relations » [Wfmc, 95].

Le processus relie donc des activités multiples effectuées par des acteurs différents utilisant des outils variés et accédant à des sources d'informations diverses.

Une application workflow connaît le processus à appliquer. Elle sait décomposer ce processus en activité, les activités en tâches élémentaires et affecter chaque tâche à un rôle, selon les règles de gestion en vigueur. L'ensemble des tâches doit être exécuté selon un ordre défini, par les bons rôles au bon moment, afin de réaliser correctement le processus. On peut considérer le workflow comme un mode de division du travail entre rôles, qui facilite la coordination des actions entre elles.

3.3. Concepts

Les principaux concepts d'un processus workflow sont:

- **Activité :** est un ensemble partiellement ordonnée de tâches. L'activité est l'objet élémentaire de gestion qui permet de maîtriser le management d'un processus [Saadoun, 02].

Une activité implique le déroulement d'un ensemble d'actions (ou opérations) visant à remplir une finalité globale. Une activité est un enchaînement de tâches élémentaires, déclenchées en amont par une demande et aboutissant à un résultat attendu. L'activité est le niveau de décomposition d'un processus qui permet de formaliser une procédure, c'est-à-dire une description structurée de la suite des opérations réalisées pour accomplir l'activité.

- **Tâche :** est une unité de travail, accomplie par un rôle pendant une période bien déterminée. La tâche (ou opération) correspond à un niveau de décomposition de l'activité. Comme nous l'avons précisé, une activité est un enchaînement de tâches élémentaires.

Dans un workflow, on peut distinguer des tâches manuelles et des tâches automatisées. Ces deux catégories de tâches sont prises en compte dans la conception et la réalisation d'un workflow. Les tâches dites « manuelles » ne sont pas contrôlables par l'application de workflow. Par exemple, l'appel d'un fournisseur au téléphone pour négocier un prix qui sera ensuite enregistré dans l'édition d'une commande, sera considéré comme une tâche « manuelle » alors que l'édition de la commande sera considérée comme une tâche automatisée.

- **Rôle :** un rôle est une collection d'acteurs, collaborant afin d'atteindre un objectif commun.

- **Acteur :** est une personne, ou un outil.

Le workflow est un ensemble de procédures, où les tâches et les informations sont échangées entre participants selon des règles prédéfinis pour accomplir ou attribuer à un objectif global, donc la finalité de ces procédures étant connue.

3.4. Les trois R du workflow

Trois éléments caractérisent fondamentalement tout workflow. Ce sont les routes, les règles et les rôles.

L'image des "3R" énoncée par Ronni Marshak [Levan, 99b], analyste connu des technologies groupware et workflow, résume bien les principales caractéristiques d'une application workflow.

•**ROUTES**: les routes déterminent des chaînes d'activités plus ou moins complexes (parallélisme, séquencement...) qui forment des cheminements le long des quels se ce déplacent les objets de gestion [Levan, 99b].

Les routes définissent les itinéraires du workflow, c'est-à-dire la synchronisation des activités et les chemins des flux informationnels qui s'appliquent en fonction des règles plus ou moins prédéfinies.

•**REGLES**: les règles regroupent les informations concernant les tâches à réaliser pour accomplir une activité (règles de gestion, formulaire, données, applications) [Levan, 99b], c'est-à-dire les tâches et les éventuelles nécessaires à leur réalisation. Ces règles permettent de fixer les principes de gestion, les données et les applications externes de workflow qui seront souhaitées.

•**ROLES**: les rôles définissent les compétences nécessaires pour assurer la responsabilité d'activités à accomplir et de résultats à obtenir.

3.5. Typologie du workflow

Il existe plusieurs types de workflow. Ces différents types dépendent des objectifs et des besoins des organisations. D'ailleurs, beaucoup de grandes entreprises utilisent plusieurs produits de workflow pour des types de workflow différents.

Tous les workflow ne mettent pas en œuvre des fonctionnalités identiques. Ils n'ont également pas tous la même architecture technique. Ces applications tentent en effet de répondre à des besoins organisationnels variés et spécifiques à chaque entreprise. Il est donc normal d'établir une typologie des applications workflow en fonction des différents types de processus [Saadoun, 96].

L'approche la plus connue est basée sur des critères fonctionnels qui établiront une classification centrée sur le workflow dépendant de la fonction que requiert le processus.

Cette typologie va différencier les applications workflow devant automatiser des procédures de production, dont les règles peuvent être définies à l'avance, de celles devant automatiser des procédures d'exception, dont il n'est pas toujours possible de définir les règles à l'avance.

3.5.1. Workflow de production

Correspond aux processus de base de l'entreprise. Les activités quotidiennes qui s'y rattachent subissent un peu de changement dans le temps et les transactions sont totalement répétitives [Blau,01]. Citons par exemple les processus de production de contrats d'assurance, la gestion de réclamations clients, ...etc.

Donc les wokflows de production s'appliquent aux processus opérationnels répétitifs et critiques, en fait aux processus métiers de l'entreprise ou du groupe qui les utilise, et qui requiert une forte productivité et une réponse rapide du système.

Ce type de workflow est caractérisé par un cadre procédural formel s'appliquant à toutes les activités et rôles impliqués par le processus en question.

3.5.2. Workflow coopératif (collaboratif)

C'est un workflow "intelligent" qui permet à une équipe de modéliser un processus de travail, d'en fixer les règles, d'exploiter directement l'application et, de façon itérative, de faire évoluer le processus et ses règles de gestion en fonction des transformations des modes opératoires, donc il gère des procédures évoluant assez fréquemment, et liées à un groupe de travail restreint dans l'entreprise.

Ces nouvelles applications de workflow (coopératifs) essayent d'allier la complexité des processus et la souplesse organisationnelle souhaitée par les utilisateurs [Frey, 99].

3.5.3. Workflow ad hoc

Ils s'appliquent aux procédures d'exception, qui n'arrivent qu'occasionnellement ou même qu'une seule fois dans la vie de l'entreprise (par exemple workflow mis en place pour la fusion de deux entreprises).

Ces processus peuvent toutefois représenter des enjeux critiques pour l'entreprise mais, généralement, ils sont liés à des routines administratives. Les workflow ad hoc s'appliquent surtout à des processus qui se rapprochent des projets [Blau, 01]. Ils font appel aux moyens de communication qui permettent l'ajustement mutuel des individus impliqués.

Les applications de workflow ad hoc sont souvent associées à des outils courants tels que tableurs et traitements de textes. Les documents étant éventuellement transportés par une messagerie, les workflows ad hoc sont beaucoup plus flexibles que les workflows de production car ils sont par nature communicants [Blau, 01].

3.5.4. Workflow administratif

Ils s'appliquent aux processus qui sont bien définis et dont les exigences de performance sont moindres que celles des workflows de production, de part leur capacité de traitement inférieur.

Le workflow administratif correspond à des processus de routages de formulaires simples. Il s'agit alors d'automatiser la manipulation de formulaires électroniques, en remplacement des imprimés traditionnels, entre plusieurs personnes intervenant sur leur traitement et suivant des procédures pré définies. Les applications de workflow administratif associent généralement un gestionnaire de formulaires (conception et utilisation) à un moteur de messagerie. Par exemple la gestion d'autorisation de dépenses, la gestion de frais de mission [Frey, 99].

Le tableau suivant résume les différences entre les quatre différents types d'applications workflow :

Production	Administratif	Coopératif	Ad-hoc
Haute capacité de traitement	Capacité de traitement inférieure moins que pour le Workflow de production.	La capacité de changer la définition d'un processus est essentielle.	Facilité d'utilisation et d'apprentissage sont très importantes.
Employés travaillant à plain temps sur des activités courtes	Un grand nombre d'employés peut être impliqué.	Fournir une voie structurée pour travailler ensemble.	La modification dynamique et rapide des processus est essentielle.
Processus formels avec peu de variation.	Une variété de processus peut exister dans le même système	Les processus sont moins rigides.	Facilité de déploiement.

Tableau I-2: Les différences entre les différents types d'applications workflow [Blau, 01]

3.6. Système de gestion de workflow (SGWF)

Le système de gestion de workflow est un système complet qui sert à définir, gérer et exécuter des procédures (workflow) en exécutant des programmes dont l'ordre d'exécution est prédéfini dans une présentation informatique de la logique de ces procédures [Levan, 99b].

Il est capable d'interpréter la définition d'un processus, de gérer la coordination des participants et d'appeler des applications externes [Blau, 01].

3.6.1. Modèle de référence d'un système de gestion de workflow

La WfMC établit un modèle afin de définir les principales composantes du système de gestion de workflow et pour créer un cadre d'interopérabilité et de référence. Pour obtenir l'interopérabilité entre plusieurs produits de workflow, il faut définir des standards d'interfaces et d'échange de données.

Avant de présenter ce modèle de référence, nous présentons la WfMC.

3.6.1.1. La WorkFlow Management Coalision (WfMC).

La WfMC est un groupement de compagnies qui se sont joint pour définir de tels standards, il a été créé en août 1993 [Wfmc, 95]. La WfMC est une organisation internationale à but non lucratif qui regroupe des éditeurs, des utilisateurs et des experts dans le domaine du workflow [Wfmc, 95].

La mission de la Coalition est de promouvoir l'utilisation du workflow grâce à la définition de standards portant sur la terminologie workflow, l'interopérabilité et la connectivité entre les produits workflow. Regroupant en 1997 près de 190 membres [Levan, 99b], la Coalition s'impose aujourd'hui comme la principale entité de standardisation et de référence pour un marché workflow en très forte expansion.

La WfMC compte aujourd'hui un grand nombre de membres, principalement les vendeurs de solutions workflow, mais aussi des analystes et des utilisateurs, parmi les membres, on trouve Bull, IBM, Microsoft et Oracle.

3.6.1.2. Terminologie

1)Acteur du workflow (workflow participant) : un acteur du workflow est une ressource (programme automatique, être humain ou être humain utilisant un programme ayant une interface utilisateur) qui exécute une activité [Levan, 99b].

2)Corbeille (WorkList ou WL) : l'interaction d'un workflow avec ses divers participants se fait par des worklist (WL). La WfMC définit un WL comme étant une liste d'activités à exécuter par un participant spécifique (ou parfois par un groupe de participants).

3)Business Object (BO): lorsque l'exécution des opérations d'un workflow se fait de manière automatique, des composants spécialisées, appelées Business Objects (BO), sont introduites pour prendre en charge ces opérations et assurer leurs réalisations. Un BO set décrit par ses attributs, son comportement, ses relations avec l'environnement intérieur et extérieur et ses contraintes [Levan, 99b].

La figure suivante présente les composants de base du modèle de référence du SGWF de workflow ainsi que les interfaces entre ces composants.

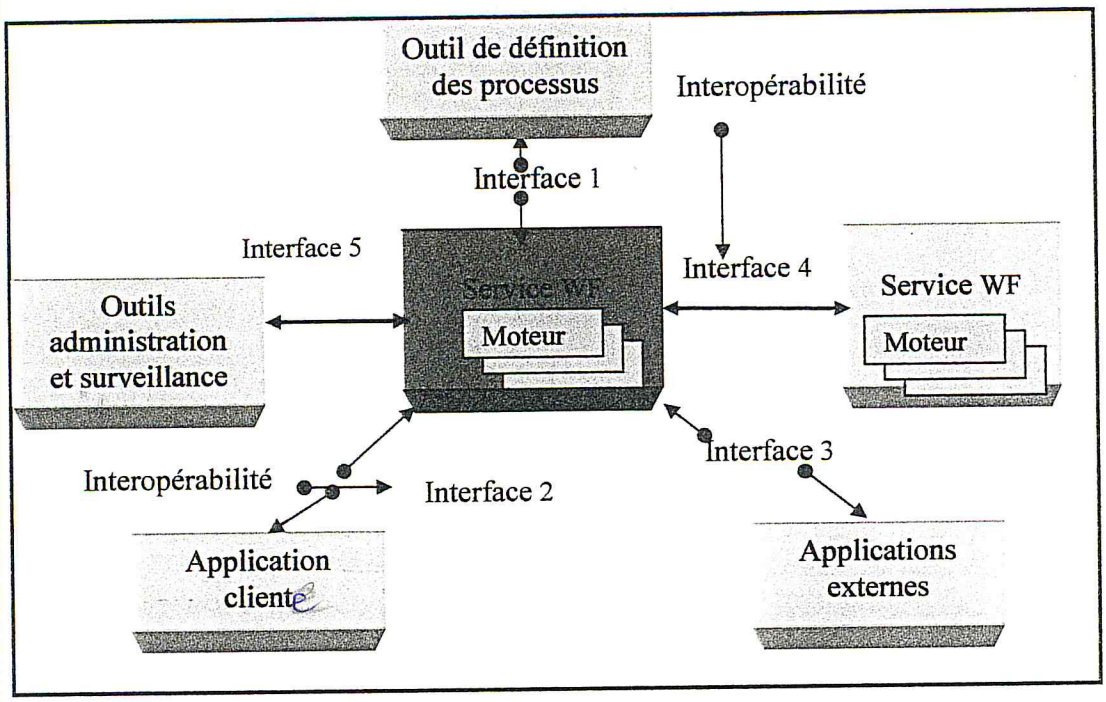


Figure I-2: Modèle de référence d'un SGWF [Levan, 99b]

Le modèle de référence d'un SGWF définit cinq composants :

1)Outils de construction/définition des processus : ce sont des outils utilisés pour analyser, modéliser et décrire un processus dans une notation abstraite la logique de fonctionnement des processus. Le résultat de ces outils est une définition des processus qui sera interprétée par le moteur workflow.

2)Service WF[8] centre nerveux du système workflow : ce service consiste en un ou plusieurs moteurs de workflow, il est capable d'exécuter un ou plusieurs workflow. Il peut impliquer plusieurs produits workflow différents [Blau, 01]. Le serveur workflow a deux interfaces (2 et 3) qui représentent d'un côté le lien vers les applications clientes du workflow et d'un autre côté les applications appelées par le workflow.

3)Application cliente : L'application cliente est chargée d'organiser le travail pour le compte d'un utilisateur. Cette application cliente peut appeler des applications et des outils logiciels nécessaires à l'accomplissement des tâches. C'est une application graphique destinée à la gestion des WL d'un usager sur le serveur workflow.

4)Applications externes : Les systèmes de gestion de workflow doivent communiquer avec toutes les applications externes nécessaires à l'accomplissement des tâches, pour cela des ressources sont utilisées par le serveur workflow pour compléter les activités des processus.

5)Outils d'administration et de surveillance : ce sont des outils utilisés pour suivre l'exécution des processus. Ils permettent d'observer l'état d'avancement du processus.

Le modèle de référence d'un workflow dispose de cinq interfaces reliant les composants précédents entre eux :

1)Interface 1(serveur–concepteur) : cette interface est prévue dans le modèle de référence de la WfMC permettant d'importer/exporter des définitions de processus. Elle définit un format

[8] **WF** :WorkFow

commun pour l'échange des spécifications des processus statiques entre l'outil de définition des processus et le serveur workflow.

2)Interface 2(client-serveur): Cette interface est celle du client à travers laquelle le moteur de workflow interagit. Elle supporte les interactions entre l 'application client du workflow et le serveur workflow. Ces interactions incluent les WL.

L'application cliente peut être un produit tiers (une messagerie par exemple), une application spécifique ou elle peut également être partie intégrante du système de gestion de workflow. Pour cette raison, il faut avoir une interface standard bien déterminée pour garantir une communication aisée et pour apporter un ensemble de fonctions de connexion entre le serveur workflow et l'application cliente.

Cette interface permet également à une application client d'un vendeur d'interagir avec le serveur d'un autre vendeur (interopérabilité workflow /application usager).

3)Interface 3(invocation d'application) : cette interface permet au moteur de workflow d'appeler une application spécifique d'une activité donnée pour exécuter une certaine tâche [Blau, 01]. Cette interface est intégrée au module serveur et ne demande donc pas d'action particulière de l'utilisateur.

4)Interface 4(serveur –serveur) : (interopérabilité workflow / workflow) : Un des grands objectifs de la WfMC était de définir des standards permettant que différents systèmes de gestion de workflow puissent travailler ensemble quand il le faut. Les produits workflow n'ont pas tous le même degré de complexité. Il en existe qui sont de simples outils de routage de données (workflow ad hoc), et il en existe d'autres qui font la gestion de processus complexes et évolutifs (workflow coopératif). Tous ces produits satisfont des besoins propres aux utilisateurs et tirent des avantages dans leur domaine. Il faut donc que ces différents produits puissent interagir.

Pour garantir une interopérabilité entre différents systèmes de gestion de workflow, la WfMC a identifiée 8 niveaux d'interopérabilité qui vont du point d'aucune interopérabilité (les produits ne sont pas du tout compatibles entre eux) jusqu'à l'interopérabilité complète (les produits disposent d'un environnement de travail commun) [Blau, 01].

5)Interface 5(surveillant-serveur) : Il s'agit de définir un standard d'interface permettant à un outil d'administration et de pilotage de travailler avec n'importe quel moteur de services workflow [Saadoun, 96]. Tout d'abord cela permettra d'obtenir une vision complète de l'état d'un workflow cheminant à travers toute une organisation, indépendamment des systèmes workflow mis en œuvre. Ensuite, cela permettra de choisir le meilleur outil d'administration et de pilotage en fonction des ses besoins et objectifs.

3.6.2. Les phases de développement d'un SGWF

Les différentes phases par lesquelles doit passer la construction d'une application de workflow sont:

3.6.2.1. La phase d'analyse

Cette phase permet la définition, et peut-être la modélisation du processus de workflow et de ses activités constituantes [Levan, 99b]. Durant cette phase le processus de métier est transformé du monde réel vers une définition informatique en utilisant une ou plusieurs techniques d'analyse de modélisation et de définition de systèmes.

La définition de processus peut être exprimé sous une forme textuelle, graphique ou dans une notation d'un langage formel.

3.6.2.2. La phase de construction

Elle consiste, à partir des modélisations de processus issus de la phase précédente, à formaliser les procédures résultantes au sein d'un outil informatique [Levan, 99b], et à définir l'ensemble des conditions nécessaires à son bon fonctionnement, et à son intégration dans l'informatique existante. Tous les produits de workflow possèdent un module gérant cette phase.

3.6.2.3. La phase d'exécution

C'est la phase pendant laquelle les procédures sont exécutées et les tâches traitées, c'est également pendant cette phase que les statistiques, fondamentales pour le suivi de tout processus, sont générées [Levan, 99b].

Des outils d'administration doivent également exister afin de pouvoir intervenir à tout moment sur les procédures elles-mêmes en cas de problème. Bien entendu tous les produits de workflow intègrent ce module.

La Figure I-3 montre les phases de construction d'un système de gestion de workflow:

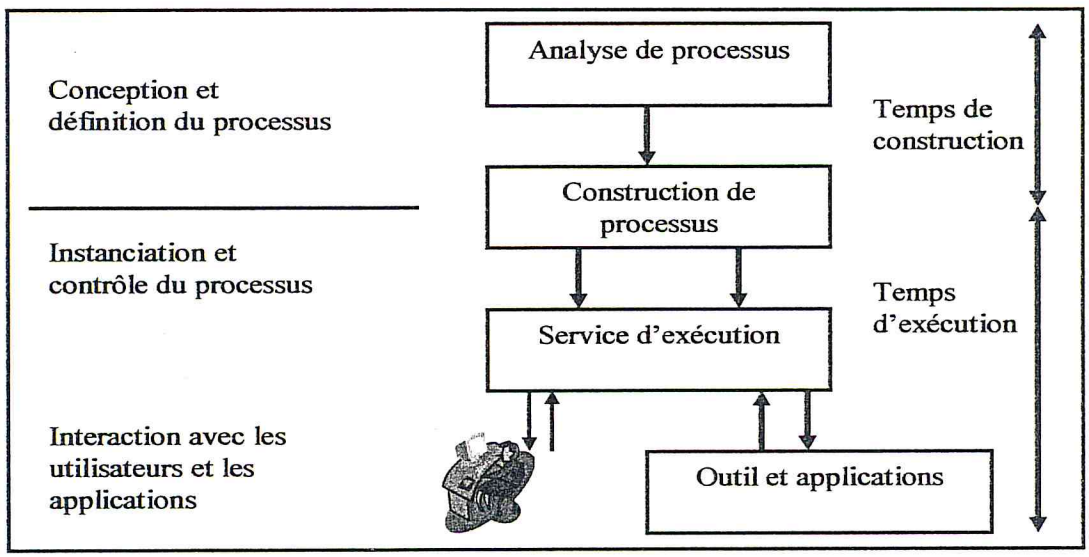


Figure I-3: Système de Gestion de workflow [Frey, 99]

3.7. Implantation de workflow

Cette partie va montrer la démarche à effectuer lorsque nous nous trouvons en face d'un projet de workflow. Il existe donc deux grandes étapes dans un projet de workflow dans une organisation. D'abord, les processus doivent être modélisés, en recourant à une technique de modélisation et ensuite le workflow doit être implémenté en recourant à cette modélisation des processus établie auparavant.

3.7.1. Modélisation des processus

La modélisation des processus vise tout d'abord à représenter sous forme graphique, le fonctionnement de l'entreprise en utilisant un langage spécifique [khochafiane&al, 98]. Il est important d'arriver à une modélisation qui est suffisamment pertinente pour qu'on puisse se baser sur elle dans des buts d'amélioration de processus.

Il existe toujours différents éléments de base d'un processus devant être modélisés :

- l'activité symbolisant une étape du processus.
- le rôle accomplissant une activité.

- la route représentant la transition entre les activités.

Une activité de modélisation commence donc toujours par la description de l'existant.

Comment l'entreprise fonctionne ? Qui fait quoi ? Comment est ce que les choses sont faites ?

3.7.2. Implémentation du workflow

L'implémentation d'un modèle de processus dans un SGWF, se fait après, sur la base de la modélisation des processus. Cette dernière, consiste à implémenter une représentation informatique du modèle de processus en utilisant un outil informatique, cet outil est nommé outil de définition de workflow (module Build Time du SGWF) [Levan, 99b]. Ce modèle contient toutes les définitions de processus nécessaires avec les informations importantes.

L'outil de définition peut être totalement indépendant du SGWF. A ce moment, il existe deux possibilités: soit est utilisée une technologie intégrée, ce qui fait que l'outil peut tout simplement exporter toutes les données dans le SGWF, soit les données doivent d'abord être traduites afin de définir le processus, dans le cas où un outil de définition indépendant est utilisé. La tendance actuelle du marché est plutôt que les outils de modélisation de processus et des outils de workflow se rapprochent où les modélisations peuvent être directement traduites [Levan, 99b], c'est-à-dire une définition est généré automatiquement.

L'implémentation du workflow permet de définir la logique déterminant dynamiquement les itinéraires d'un processus préalablement modélisé [Blau, 01]. Les 3R de Ronni Marshak doivent être respectés. Il faut donc définir trois variables différentes dans une implémentation de workflow:

- **Les rôles** devant être définis indépendamment des individus réellement impliqués dans l'entreprise.
- **Les règles** décrivant les conditions d'exécution des activités, en intégrant les applications extérieures au workflow (traitement de texte, tableur…etc.)
- **Les routes** assurant la liaison entre les activités et les acteurs.

Pour faciliter la compréhension, nous montrons l'implémentation de manière graphique sur la Figure I-4. L'ensemble des tâches d'implémentation se fait dans le module de Build Time du système de gestion de workflow. Le Run Time de son côté gère les instanciations et le contrôle des processus ainsi que les interactions avec les ressources et applications.

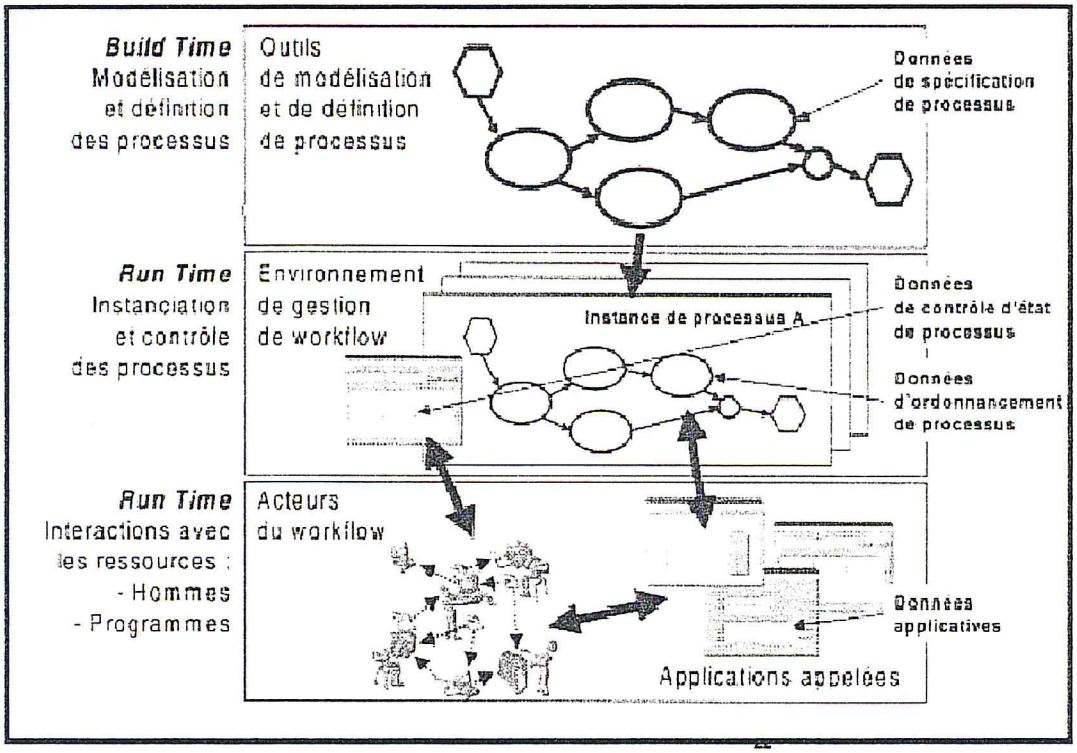


Figure I-4: Le Build et le Run Time d'un système de gestion de workflow [Levan, 99b]

3.8. Avantages des applications du workflow

Les systèmes automatisés de workflow peuvent offrir de nombreux avantages fonctionnels, dont voici les plus courants :

•Accroissement de la productivité

La plupart des organisations se doivent de maximiser l'utilisation de leurs ressources et de veiller à consacrer leur temps aux activités qui contribuent directement à leurs objectifs d'affaires. Grâce aux systèmes de workflow, les tâches sont saisies, mises en ordre de priorité et acheminées vers la personne ou la fonction la mieux placée pour les accomplir. Les règles et les instructions décrivant la tâche à accomplir accompagnent l'acheminement. Le workflow permet également de réduire considérablement le traitement des erreurs, le contrôle de la qualité et les reprises.

•Diminution des cycles de travail

Le workflow offre les moyens de contrôler la durée du cycle de vie d'un processus métier, de son déclenchement jusqu'à son achèvement [Frey, 99]. Il permet de supprimer les délais de traitement superflus, comme le temps passé à transférer des documents et des dossiers de papier. Il n'y a plus de documents perdus ou mal classés.

•Amélioration du contrôle

Les alternances du volume des activités peuvent être facilement contrôlés pour régler les problèmes à mesure qu'ils surviennent. Les systèmes de workflow peuvent fournir, à tout moment au cours d'une journée, un "instantané" des tâches non accomplies et produire des mises à jour sur la distribution des tâches et les activités en cours [Frey, 99].

•Amélioration du service à la clientèle

La clé d'un bon service à la clientèle est une réponse rapide et efficace aux demandes des clients. Un système de workflow peut permettre de traiter toutes les demandes de façon rapide et de mettre en tête de file les tâches qui remplissent certaines conditions prédéterminées (comme une demande de votre meilleur client). De plus, le logiciel de workflow assemble tous les éléments et renseignements nécessaires à l'accomplissement de la tâche, y compris des données sur les clients et leurs contrats commerciales.

•Avantage concurrentiel

La mondialisation des marchés accroît la concurrence. Grâce à l'avancement des technologies et des communications, n'importe qui peut faire affaire avec un client ou un fournisseur donné, quelle que soit la distance [Frey, 99]. Pour être concurrentiel, il faut pouvoir réagir rapidement aux changements du marché et aux nouvelles demandes des consommateurs. Un système de workflow permet à l'organisation de réagir plus rapidement.

4. Conclusion

Le workflow représente un type particulier d'application de groupware. Son rôle principal est d'assister les personnes impliquées dans l'accomplissement des activités et des tâches d'un processus métier. Chaque personne impliquée dans le processus accomplit tout ou partie de ses tâches à partir de son ordinateur connecté au réseau Intranet/Internet. L'ordonnancement des activités et des tâches est mémorisé par le moteur de workflow qui gère les interactions entre les participants. Ceux-ci peuvent ainsi travailler ensemble malgré certains obstacles de distance ou de temps.

Afin de mettre en place une application de workflow, nous devrons procéder par une étape primordiale qui nous permet de définir le processus à exécuter dans un formalisme adéquat. La présentation des formalismes utilisés fera l'objectif du prochain chapitre.

CHAPITRE II.
EXPRESS ET UML

1. Introduction

Dans ce chapitre nous présentons les langages que nous utilisons pour le développement de l'outil : EXPRESS et UML. Le langage EXPRESS est utilisé pour la création de la base de données, et UML pour le développement de l'outil.

Le langage EXPRESS est le résultat du projet international STEP (STandard for the Exchange of Product model and data-ISO 10303-11) [Spidy, 94], c'est un langage de spécification formelle de données c'est à dire qu'il permet de décrire sous forme traitable par machine des modèles de données. C'est aussi un formalisme de modélisation conceptuelle [Ait Ameur, 02].

L'Unified Modeling Language (UML) est un formalisme de modélisation orienté objet. Comme son nom l'indique, ce langage est le résultat d'une longue maturation. Il est la fusion de plusieurs langages de modélisation, issus notamment de la méthode de Grady Booch , de la méthode OOSE[9] de Ivar Jacobson, la méthode OMT[10] de James Rumbaugh, et de La méthode BOOCH de Booch [Fannader &all, 00].

Afin d'étudier ces langages, nous avons organisé ce chapitre comme suit :

Nous commençons ce chapitre par une présentation les notions de l'environnement orienté objet (l'objet, la classe, l'héritage, …etc.), et de modélisation (modèle, formalisme, méta modèle…etc.).

Ensuite, nous présenterons en deuxième partie une étude du langage EXPRESS; nous montrerons dans cette partie les différents concepts manipulés par ce langage, ainsi que sa notation graphique, EXPRESS-G.

Dans une troisième partie, les différents concepts sur lesquels UML s'appuie seront traités, en présentant ainsi ses différents diagrammes.

---

[9] **OOSE**: Object Oriented Software Engineering
[10] **OMT**: Object Modeling Technique

2. Les concepts fondamentaux de l'approche orienté objet

Parmi les concepts fondamentaux de l'approche orienté objet, nous notons les objets, les classes, l'encapsulation, l'héritage, le polymorphisme et l'agrégation.

2.1 . L'objet : est une abstraction informatique d'une entité du monde réel caractérisée par une identité, un état et un comportement [Gardarin, 02].

Formellement un objet est l'unité formée d'un état et un comportement. L'état d'un objet est formé des valeurs instantanées de ses attributs, ainsi que le comportement regroupe les compétences de l'objets. Ce comportement est décrit par les méthodes déclenchées par des stimulations externes appelées messages. Pour identifier les objets, chaque objet est caractérisé par un identificateur unique *oid*[11].

2.2 . La classe : est la description d'un groupe d'objets ayant des propriétés similaires, et un comportement commun [Muller, 97]. C'est une implémentation d'une ou plusieurs interfaces sous la forme d'un moule permettant de spécifier un ensemble de propriétés d'objets (attributs et opérations) et de créer des objets possédant ces propriétés [Gardarin, 02].

2.3 . L'abstraction : est la faculté des humains de se concentrer sur l'essentiel et d'oublier les détails [Muller, 97], c'est à dire consiste à se concentrer sur les aspects essentiels, inhérents, d'une entité et à ignorer les propriétés accidentelles. Dans le développement des systèmes, cela signifie mettre l'accent sur ce qu'est un objet et sur ce qu'il fait, avant de décider comment il doit être implémenté [Pillou, 02].

2.4 . L'encapsulation : est un occultation des informations contenues par un objet [Muller, 97], c'est à dire un mécanisme consistant à rassembler les données et les méthodes au sein d'une structure, en cachant l'implémentation de l'objet, et par conséquent en empêchant l'accès aux données par un autre moyen que les services proposés [Pillou, 02]. L'encapsulation permet donc de garantir l'intégrité des données contenues dans l'objet.

2.5 . L'héritage : est une transmission automatique des propriétés d'une classe de base vers une sous-classe [Gardarin, 02]. Il est aussi définit comme une relation entre classes qui permet le partage de propriétés définies dans une classe, principale technique de réalisation de la généralisation [Muller, 97].

2.6 . Agrégation : est une association entre deux classes exprimant que les objets de la classe cible sont des composants de ceux de la classe source [Gardarin, 02]. C'est un autre type de relation entre deux classes qui traduit par les relations « **Est composé de ...** » ou « **Possède ...** » ou encore « **a ...** » [Pillou, 02]. Donc l'agrégation est une relation qui permet de décrire un objet composite en terme d'objets qui le constituent. L'une des caractéristiques principale de l'agrégation est sa **cardinalité.**

2.7 . Le polymorphisme : est la faculté pour une opération d'avoir différentes signatures avec un code spécifique attaché à chaque signature [Gardarin, 02]. Le polymorphisme autorise qu'une opération soit applicable sur des objets de types différents, le code à exécuter étant déterminé par le type de paramètres de la méthode. Il permet à une méthode d'adopter plusieurs formes sur des classes différentes.

3. Des notions de modélisation

Dans cette section, nous proposons de définir et positionner certaines notions de modélisation, afin de lever toute ambiguïté terminologique dans la présentation que nous faisons des techniques de modélisation de donnée dans ce chapitre où dans les chapitres suivants.

---

[11] **oid**: Object IDentifier

3.1 . Modèle : un modèle tel que défini par N.Kettani [kettani & al, 01] est une abstraction de la réalité qui, pour un domaine, est prise comme une représentation d'une classe de phénomènes plus ou moins habilement dégagés de leur contexte par un observateur, pour servir de support à l'investigation et/ou à la communication. Un système est décrit par plusieurs modèles, chacun d'entre eux définit un aspect spécifique de ce système (statique, dynamique et fonctionnel).

Cette représentation est exprimée dans un langage de représentation qui peut être formel (ayant une syntaxe et une sémantique bien définies), semi-formel (notation graphique normalisée) ou informel (description en langage naturel).

3.2 . Méta modèle : décrit de manière formelle les éléments de modélisation, la syntaxe et la sémantique de la notation qui permet de le manipuler [Muller, 97].

3.3 . Langage : est un ensemble de constructions qui permettent de décrire formellement les spécification du système d'information élaborées aux différents stades du processus de conception. Les éléments constituants d'un langage de modélisation sont : des concepts, une syntaxe et une sémantique [Muller, 97].

3.4 . La modélisation : consiste à créer une représentation informatique des éléments du monde réel auxquels on s'intéresse, sans se préoccuper de l'implémentation, ce qui signifie indépendamment d'un langage de programmation [Muller, 01].

3.5 . Formalisme : ce terme est utilisé pour désigner un langage de modélisation. Il désigne un ensemble d'éléments de modélisation auquel est associe une représentation souvent graphique, qui permet de manipuler aisément les modèles et de communiquer l'information entre les différents intervenants d'un projet informatique [Muller, 01].

3.6 . Diagramme : une représentation graphique d'une collection d'éléments de modèle, le plus souvent présentée comme un graphe connexe d'arcs [kettani & al, 01]. Un diagramme n'est pas un modèle mais une représentation graphique de quelques éléments du modèle, de ce fait un diagramme, n'est qu'une projection du modèle visant un aspect bien précis du modèle, et par conséquent un modèle est une collection de plusieurs diagrammes.

4. Le langage EXPRESS

Le langage EXPRESS fut développé dans les années 80 dans le cadre du projet STEP. Son objectif initial est la normalisation des modèles de données pour pouvoir les échanger et les réinterpréter sans difficultés [Spidy, 94]. Le langage EXPRESS est utilisé pour exprimer formellement ces données.

Ce sont Bernd WENZEL et Douglas SHENCK qui ont mis au point ce langage. A partir de cette version syntaxique d'EXPRESS, Peter WILSON a développé une version permettant de représenter graphiquement les données: EXPRESS-G [Spidy, 94].

Le langage EXPRESS est basé sur la précision du modèle et spécialement sur les contraintes que doivent respecter les données pour être acceptées comme semblable au modèle, a fin d'assurer la fiabilité de l'information représentée [Ait Ameur, 02]. Ce n'est pas seulement une notation permettant la *modélisation* des données, (c'est à dire une représentation simplifiée, peut-être ambiguë). C'est également un formalisme de spécification, c'est à dire qu'il permet une description complètement non ambiguë et traitable par machine.

Le format textuel en fait toute sa puissance d'expression par rapport aux autres formalismes: il est compilable et génère des modèles de données pour les langages C++, JAVA,... etc.

4.1 . Concepts fondamentaux

Le langage EXPRESS est un langage de modélisation de données orienté objets possédant les grandes caractéristiques des langages à objets [Ait Ameur, 02] : abstraction, héritage et polymorphisme. Une spécification EXPRESS est définie par plusieurs schémas, chaque schéma définit un ensemble d'entités, qui représentent les objets à modéliser. Chaque entité est définie par un ensemble de caractéristiques appelées attributs. Chaque attribut possède un domaine de valeurs licites (type de données). EXPRESS permet de contraindre ces domaines grâce à des contraintes d'intégrité.

Dans cette section, nous présentons les principaux concepts qui proviennent de ce langage (entités, attributs, contraintes, fonctions et procédures... etc), à travers des exemples, et afin de bien comprendre le langage EXPRESS, nous utilisons une notation, qu'est inspirée de la célèbre notation BNF[12].

::=	-- définition
|	-- séparateur d'alternatives
SYMBOL	-- mot clé
symbol	-- symbole terminal
<symbol>	-- symbole non terminal
{term}	-- term peut apparaître 0 ou n fois
[term]	-- term peut apparaître 0 ou 1 fois
' term'	--term est écrit obligatoirement

Figure II-1 : Les symboles de la notation utilisée

Toutes les syntaxes des concepts du langage EXPRESS présentés dans la section suivante sont inspirées du manuel de référence de l'EXPRESS [Spidy, 94].

4.1.1 .Les schémas

Un schéma est une unité de modélisation. Il constitue un conteneur dans le quel on déclare tous les concepts définis en EXPRESS: les constantes, les types, les entités et les contraintes Un schéma est déclaré par le biais du mot clé SCHEMA.

La syntaxe
<decl _schema> :: = SCHEMA id_schema ';' <corps_schema> END_SCHEMA ';'

Figure II-2 : La syntaxe d'un schéma

- **Exemple : SCHEMA** a1 ; ..... **END_SCHEMA** ;

4.1.2 .Les entités

Une entité correspond à ce qui dans le monde réel possède une existence effective et autonome [Ait Ameur, 02]. En EXPRESS une entité est un couple formé d'un identificateur et un ensemble de propriétés (attributs).

Les propriétés d'une entité se décrivent sous la forme d'attributs. La déclaration d'une entité est exprimée par le mot clé **ENTITY**.

[12]BNF :Forme Backus-Naur.

La syntaxe
< decl _entite > ::= <tete_entite>< corps _entite > END_ENTITY ';'
< tete_entite> ::= ENTITY id_entite ';'
<corps _entite> ::= {attr_ explicit} [<derive_clause>] [<inverse_clause>]
[<unique_clause>] [< where_clause>] .

Figure II-3: La syntaxe de l'entête et le corps d'un schéma

Les attributs possèdent plusieurs statuts dans leurs déclarations.

- **Libre**: indépendant vis à vis des autres attributs et sa valeur (obligatoire) dépend d'une saisie utilisateur.
- Optionnel (**OPTIONAL**): la présence de valeur n'est pas obligatoire. Un attribut déclaré optionnel dans une super-classe, reste optionnel dans toutes les sous-classes sauf s'il est redéfini.
- Dérivé (**DERIVE**): la valeur de l'attribut est calculée par une fonction. Il est hérité par toutes les sous-classes de la classe où il est déclaré.

La syntaxe
<derive_clause> = DERIVE <attr_derive> { <attr_derive> }
< attr_derive> = id_attribu ':'type_ base ':=' expression ';'

Figure II-4 : La syntaxe de la clause DERIVE

- **Exemple**

```
ENTITY person;
        Nom: STRING ;
        Born: date;
       Prénom_2 :OPTIONAL STRING;
        DERIVE
          Age: INTEGER := years_since(born);--  years_since:est une fonction.
END_ENTITY;
```

Une entité peut hériter d'une ou plusieurs entités. Un sous type hérite de toutes les propriétés et les contraintes locales de ses super-types. De même les contraintes globales qui s'appliquent au super-type, s'appliquent également sur les sous-types.

Les mots clés **SUPERTYPE, SUBTYPE** permettent de définir, respectivement, la super-classe et la sous-classe. Une classe abstraite est référencie par le mot clé **ABSTRACT**.

La syntaxe
<declaration_abstract_supertype> ::= ABSTRACT SUPERTYPE OF
<supertype_expression >';'
<supertype_expression> :: = ANDOR| ONEOF|AND'(' ref_entite{',' ref_entite}') ;'
< declaration _subtype>::= SUBTYPE OF '(' ref_entite { ',' ref_entite } ') ;'

Figure II-5 : La syntaxe de l'héritage

- **Exemple** montre l' hiérarchie des classes dans l'EXPRESS.

ENTITY personne	**ENTITY** étudiant	**ENTITY** enseignant
ABSTRACT SUPERTYPE OF	SUBTYPE OF personne;	SUBTYPE OF personne;
ONEOF (étudiant, enseignant);		
END_ENTITY;	**END_ENTITY;**	**END_ENTITY;**

4.1.3 .Les types

Le langage EXPRESS dispose d'une structure complète de types permettant la définition des domaines sans ambiguïté. En EXPRESS un type est associé à un attribut, une variable locale ou un paramètre formel, pour cela on distingue quatre catégories des types: type de base, type d'agrégat, type utilisateur et type construit.

1) Type de base

Permettent de définir les domaines des objets lexicaux (string), numériques (real, binary, integer) ou logiques, (logical, boolean).

- **Exemple** Nom: STRING ;
  Age: INTEGER ;
  Marie : BOOLEAN ;

2) Types d'agrégats

Ce sont les collections des valeurs et des objets des types de base. Les agrégations sont résumés en tableaux (ARRAY), liste (LIST), sacs (BAG) et ensemble (SET).

➢ ***ARRAY :*** est un tableau d'éléments de taille fixe, fixé par une séquence de nombres entiers.

La syntaxe
<type_array>:: = id_tab ':' ARRAY '[' born_1 ':' born_2 ']' OF <type_under> ' ; '
<type_under> :: = type_construit | type_aggregation | type_simple| ref_type .

Figure II-6 : La syntaxe du type ARRAY

- **Exemple** tab: ARRAY [1:20] of INTEGER;

➢ ***LIST*** : est une séquence d'éléments, chacun étant accessible par sa position dans la séquence. Le nombre d'éléments est variable. La définition de type de donnée peut contenir des contraintes sur le nombre d'éléments

La syntaxe
<Type_list>:: = id_list ':' LIST '[' born_1 ':' born_2 ']' OF <type_under> '; '

Figure II-7: La syntaxe du type LIST

- **Exemple** liste: LIST [0:20] OF INTEGER;

➢ ***BAG*** : est un groupe d'éléments non ordonnés ou deux éléments peuvent avoir même valeur.

La syntaxe
<type_bag>:: = id_bag ':' BAG '[' born_1 ':' born_2 ']' OF <type_under> ';'

Figure II-8: La syntaxe du type BAG

- **Exemple** Line: BAG [1:?]OF POINT;

➢ ***SET :*** un groupe d'éléments non ordonné ou chacun doit être unique.

La syntaxe
<type_set>:: = id_set ':' SET '[' born_1 ':' born_2 ']' OF <type_under> ';'

Figure II-9 : La syntaxe du type SET

- **Exemple** année : SET [1:12] OF mois ;

3) Les Types nommés

Sont des types déclarées dans une spécification formelle. Il y a deux catégories de types nommées: type ENTITY et type définie.

➢ **Le type de donnée ENTITY:** ce type permet de déclarer les entités. Il est spécifie par le mot clés ENTITY.

- **Exemple**

ENTITY personne;

Nom: STRING.

Marie : BOOLEAN ;

END_ENTITY;

➢ **Type de donnée définie :** il est déclaré par le mot clés TYPE, le nom (identificateur) de ce type est mentionné par l'utilisateur.

La syntaxe
<type_definie> ::= <tete_type>< corps_type> END_TYPE ';'.
<tete_type>::= TYPE id_type (('=' <type_under> ';') | (';' { <attr_type> })) .
<attr_type> ::= dec_attribut{ ',' dec_attribut } ':' [OPTIONAL] <type _under> ';' .

Figure II-10: La syntaxe du nouveau type

- **Exemple** **TYPE** t_nom =string[13] ;**END_TYPE** ;

Remarque: la déclaration des types peut être soumise à des contraintes.

- **Exemple**

TYPE t_age =INTEGER;
WHERE Wr1 :SELF > 0;-- le mot-clé SELF: permet, entre les mots-clés
--ENTITY et END_ENTITY, d'identifier l'entité courante.
END_TYPE .

4) Type construit

Le langage EXPRESS défini deux constructeur de type ENUMERATION et SELECTION, le premier permet de définir un type, en spécifiant sa propre plage de valeurs, et le second spécifié une union de type.

La syntaxe
<type_enumeration>:: = TYPE id_type '=' ENUMERATION OF
(id_simple{','id_simple} ') ;' END_TYPE ';'
<select_type> :: = TYPE id_type '=' SELECT '('type_nomme {',' type_nomme} ')'
END_TYPE ';'

Figure II-11: La syntaxe de type construit

• **Exemples**

```
TYPE justification = ENUMERATION OF ( droite, gauche, centré, justifié);
END_TYPE;
TYPE mention SELECT  (admet, ajourné);
END_TYPE;
```

4.1.4 .Les contraintes

Le langage EXPRESS permet de contraindre les entités en définissant le domaine de validité des attributs, ou plutôt les propriétés et les règles auxquelles les attributs doivent obéir.

1) Contraintes d'unicité

Elles vont contribuer à indiquer que l'attribut concerné doit avoir une valeur unique. C'est l'équivalent de la clé primaire dans les modèles relationnels. Ces contraintes sont introduites via la clause **UNIQUE.**

La syntaxe
< unique_clause> :: = UNIQUE <unique_regle> ';' { <unique_regle> ';' }
< unique_regle> :: = label ':' id_attribut { ','id_ attribut }

Figure II-12: La syntaxe des contraintes d'unicité

2) Contraintes de cardinalité

Le deuxième type de contraintes sur les entités est la contrainte de cardinalité. Elle permet d'exprimer le nombre d'entités d'un certain type qui référencent une entité donnée dans un certain rôle. Un attribut inverse exprime le couplage entre une entité sujet (celle à l'intérieur de laquelle l'attribut inverse est déclaré) et des entités utilisatrices qui se réfèrent à l'entité sujet par un attribut. Elle s'applique à un attribut en utilisant la clause ***INVERSE***.

La syntaxe
< inverse_clause> :: = INVERSE <attr_inverse>{ <attr_inverse> }
< attr_inverse>:: = id_attribut ':' [(SET | BAG) '[' born_1 ':' born_2 ']' OF] ref_entite FOR ref_attribut '; '

Figure II-13: La syntaxe des contraintes de cardinalité

3) Les contraintes des domaines (individuelles)

Elle contraint le domaine de valeurs d'un type ou le domaine d'une instance d'un attribut, elle est vérifiable individuellement sur chaque classe, donc ces contraintes sont des règles locales (à l'entité). Ce type de contrainte est introduit par le mot-clé ***WHERE***.

La syntaxe
< where_clause> :: = WHERE <regle_domain> ';'
<regle_domain> :: = id_simple ':' < expression_logic>.
<expression_logic> ::= expression '<' | '>' | '<=' | '>=' | '<>' | '=' | IN | LIKE expression

Figure II-14: La syntaxe des contraintes de domaine

4) Les contraintes globales

Elles sont spécifiées dans le contexte d'un schéma, elle exprime une contrainte portant sur une ou plusieurs entités d'un schéma. Elles peuvent être utilisées pour exprimer une cardinalité maximale sur le nombre des instances crées à partir d'une classe. La définition de telle contrainte se fait via la clause **RULE**.

- **Exemple**

```
ENTITY personne;
  ident: INTEGER;
  conduit: SET OF vehicule;
  um_permis: OPTIONAL STRING;
UNIQUE
     ur: ident;
WHERE
     wr1: NOT (SIZEOF(conduit) > 0)
      OR EXISTS(num_permis);
END_ENTITY;
```

```
ENTITY vehicule;
  type: INTEGER;
INVERSE
 conduit_par: SET [0:?] OF personne
    FOR conduit;
END_ENTITY;
```

```
RULE autant_personnes_que_vehicules FOR (personne, vehicule);
WHERE
     SIZEOF(personne) >= SIZEOF(vehicule);
END_RULE;
```

5) Contraintes ensemblistes

Les éléments d'une collection peuvent être parcourus un par un, et peuvent être vérifiés comme répondant à des critères particuliers. Les contraintes ensemblistes permettent de représenter les quantificateurs universel '∀' et existentiel '∃'. Ce type de contraintes est introduit par le mot clé **QUERY**, définie soit dans les clauses **WHERE**, soit dans les clauses **RULE**.

- **Exemple**

```
ENTITY etudiant ;
        Nom :STRING ;
        Where   wr : SIZEOF(QUERY(x<* note |(x<0) or (x>20))) =0;
END_ENTITY;
```

- **Remarques**

1) La fonction prédéfinie SIZEOF rend la taille d'un agrégat ou le nombre d'instances existant pour un type donné d'entités.
2) **Les Commentaires:** le langage EXPRESS a introduit des commentaires délimités par (*...*). Un commentaire sur une ligne unique aurait pu être introduit par le couple de caractères "- -". Tout ce qui suit jusqu'à la fin de ligne est alors un commentaire.

4.1.5 .Les fonctions et les procédures

Nous avons défini, le langage EXPRESS comme un langage permettant la modélisation formelle des connaissances structurelles et descriptives, mais également de certaines formes des connaissances procédurales. En effet, EXPRESS dispose d'un langage procédurale, impératif proche de PASCAL incluant les déclarations de types et structure de contrôle. Notons que EXPRESS autorise la récursivité.

On distingue deux types de fonctions et procédures :

➢ **Fonctions et procédures déclarées par l'utilisateur**

Pour déclarer une fonction on utilise **FUNCTION ....END_FUNCTION ;**

- **Exemple** soit la fonction FACT qui calcule le factoriel d'un nombre.

Function fact (x:INTEGER):INTEGER; ..... **End_function ;**

Pour déclarer une procédure on utilise la clause : **PROCEDURE.....END_PROCEDURE;**

- **Exemple**

Procedure milieu (p1, p2:point;var: result: point); ..... **end_procedure;**

➢ **Fonctions et procédures prédéfinies**

EXPRESS dispose d'un ensemble de fonctions prédéfinies. Ces fonctions permettent d'effectuer les calcules arithmétiques, mathématiques sur des données mais aussi de requérir des informations sur le type des attributs et des variables.

- **Exemple** : LOG10, LENGTH, SIN: sont des fonctions prédéfinies.

4.1.6 .La réutilisation entre schémas

Un schéma peut utiliser d'autre schémas, de sorte qu'il est possible de décrire des schémas généraux, destinés à être référencés ou spécialisés par d'autre schémas plus spécifiques [AitAmeur, 02]. La réutilisation des types, entités et fonctions peuvent s'exprimer de deux manières distinctes: utilisation ou référencé via respectivement les clauses **USE** et **REFERENCE.**

- **USE** : permet l'utilisation d'un type de données entité ou un type de définition de déclaration d'un schéma étranger.
- **REFERENCE** : résout les constantes, entités, fonctions procédures et type dans un schéma courant.

La syntaxe
< reference_clause> = REFERENCE FROM ref_schema ['(' resource_or_rename
{ ',' resource_or_rename } ')'] ';'
<resource_or_rename >:: = <ref_resource> [AS <id_rename>]
<ref_ resource>:: = ref_constant | ref_entite | ref_fonction | ref_procedure | ref_type
<id_ rename>::= id_constant | id_entite| id_fonction | id_procedure | id_type

Figure II-15: La syntaxe de la clause REFERENCE

La syntaxe
< use_clause>::=USE FROM ref_schema ['(' type_renomme{ ',' type_renomme}')']';'
< type_renomme> :: = nom_type [AS (id_entite | id_type)]

Figure II-16 : La syntaxe de la clause USE

Exemple: SCHEMA s1 ;
ENTITY e1 ;
END_ENTITY ;
END_SCHEMA ;
SCHEMA s2 ;
USE FROM s1(e1 AS e2) ;--renommage;
END_SCHEMA;

4.2 . EXPRESS-G

Nous donnons dans cette section un aperçu de la forme graphique (EXPRESS-G) du langage EXPRESS. On constate que la représentation textuelle des schémas EXPRESS est difficilement lisible [Dehaninsala, 02]. C'est pour pallier ce problème que le formalisme graphique a été élaboré. Il permet de donner une vue synthétique du modèle conceptuel et est adapté dans le cadre de la phase d'analyse d'un problème de modélisation.

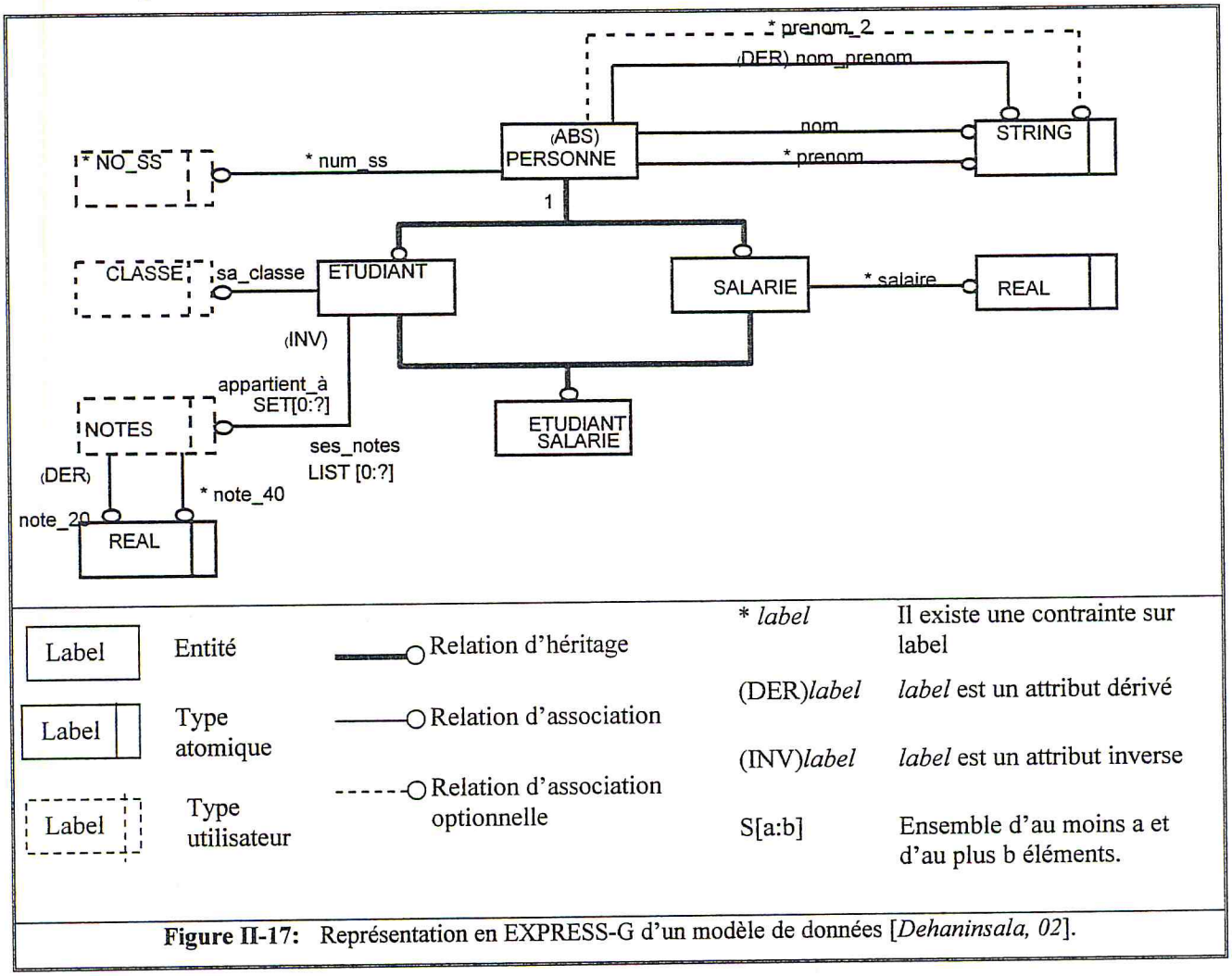


Figure II-17: Représentation en EXPRESS-G d'un modèle de données [*Dehaninsala, 02*].

4.3 . Représentation des instances : le fichier physique

Les instances d'un modèle en EXPRESS sont décrites et échangées à travers un fichier textuel nommé fichier physique [Sardet, 99]. Un tel fichier permet à deux systèmes informatiques hétérogènes d'échanger sans aucune ambiguïté une population d'instances conformes à un même modèle EXPRESS.

Chaque instance est identifiée pour un oid (Object IDentifier : #i). Elle est caractérisée par le nom de la classe instanciée. Enfin elle est décrite par la suite des valeurs de ses attributs représentées entre parenthèses. Les collections d'éléments sont définies entre parenthèses. Les attributs optionnels sont désignés par '$', et les type énumérés par une notation encadrant l'identificateur par un point à chacune de ses extrémités. Les attributs dérivés et inverses ne sont pas représentés. Un exemple d'instances du modèle de données que nous avons décrit est présenté dans la figure ci-après :

```
#1 =ETUDIANT ('03052003', /*num-ss, hérite de la classe personne*/
                'Boumahdi',        /*nom, hérite de la classe personne*/
                (15.5, 17.0),      /*notes, la listes des reels*/
                (#5, #6)) ;        /*suit, référence aux instances de cours*/
#2 =ETUDIANT ('14052003','Fareh', (12.0, 19.0), (#5, #7)) ;
#3 =ENSEIGNANT ('31052000', 'Hichem',$,(#5,#6)) ;
 #4 =COURS (.MATH. ,.A3.) ;
#5 =COURS (.INFO. , .A3. ) ;
```

Figure II-18: Un exemple de fichier physique

- **Remarque :** les attributs dérivés et inverses ne sont pas instanciés.

4.4 . Partage des instances avec l'interface logicielle SDAI[13]

En addition à la méthode d'échange définie par le fichier physique, présentée dans la section précédente, une méthode de partage d'instances entre systèmes hétérogènes a été introduite, il s'agit du SDAI. SDAI est une interface permettant :

- L'accès et la manipulation des instances d'entités définis en EXPRESS.
- L'accès simultané par plusieurs applications à plusieurs bases de données.
- L'accès à la définition EXPRESS des éléments de données qui peuvent être manipulées par une application.
- La vérification des contraintes définies dans le modèle de données EXPRESS.

Les applications souhaitant accéder à des modèles de données EXPRESS doivent implémenter ces interfaces. Dans la deuxième partie du mémoire, nous verrons la manière d'implémentation de ces interfaces.

4.5 . Les avantages du langage EXPRESS

- A la différence des nombreuses notations existantes pour modéliser des données telles que OMT ou UML qui ne sont que graphiques et donc seulement échangeable entre être humains, le fait, pour EXPRESS, de posséder à la fois une version graphique et une version textuelle rend ce langage traitable par machine.
- Un modèle EXPRESS peut être échangé entre machines différentes.
- Il peut également être traité par machine pour en vérifier la cohérence, ou pour passer automatiquement de la version graphique à la version textuelle et inversement.
- Il s'agit à la fois d'un modèle interprétable par l'utilisateur humain, et d'une spécification interprétable par machine.
- Il s'agit d'un standard international, qui est de plus en plus utilisé.
- Il possède une syntaxe et une sémantique précises, autorisant, avec des outils appropriés, une interprétation dans un langage orienté objet(C++).
- Il possède une représentation graphique qui fournit une vue synthétique du modèle pour mieux le comprendre où le concevoir c'est EXPRESS_G.

[13] **SDAI** (standard data access interface) : c'est une interface permettant de manipuler les instances du modèle d'EXPRESS.

- Il possède un système de contraintes complet (contraintes de cardinalité, contraintes ensemblistes, contraintes fonctionnelles, contraintes sur les classes...), ainsi qu'un langage procédural permettant d'enrichir celles-ci ;
- Il peut être représenté soit textuellement, soit graphiquement.

4.6 . Les inconvénients du langage EXPRESS

- Le formalisme EXPRESS ne facilite pas la communication autours de besoins, comme nous l'offre d'autre langage (UML).
- Les modèles offerts par EXPRESS ne permettent pas de couvrir l'ensemble des niveaux d'abstraction et de prendre en compte les processus métier.
- EXPRESS est un langage de modélisation de données, mais non pas de traitement chose qui constitue un handicape pour les analystes des systèmes dynamiques.
- Les modèles offerts par EXPRESS sont complexes, et difficiles à interpréter par les utilisateurs.

5. UML

Le langage UML constitue une unification des méthodes objets, tirant donc profit des avantages de chacune de ces méthodes à l'origine de l'unification, il constitue un standard pour la modélisation orientée objet.

Le modèle conceptuel d'UML comprend les notions de base génériques du langage. Il définit trois sortes de briques de base :

– Des concepts (structurels, comportementaux, annotationnels, groupements).

– Des relations (association, généralisation, les agrégations, les compositions …etc.).

Des diagrammes (statiques et dynamiques).

La section suivante a pour but de présenter le langage de modélisation de données: UML, pour remplir les objectifs suivants :

– Décrire les concepts fondamentaux de ce langage, ceux-ci étant illustrés par des exemples.

– Montrer les différentes relations entre les classes.

– Présenter l'ensemble de ses diagrammes (cas d'utilisation, classe, objet…etc.).

5.1 . Les concepts

UML supportent quatre (04) types de concepts : [kettani & al, 01]

5.1.1 .Les concepts structurels : représentés par les classes, les interfaces, les collaborations…

5.1.2 .Les concepts comportementaux: représentés par les interactions et les états des objets.

5.1.3 .Les concepts annotationnels: représentés par les notes.

Une note : est un commentaire attaché à un ou plusieurs éléments de modélisation [Muller, 97].

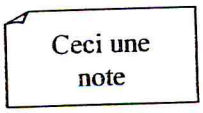


Figure II-19: Représentation d'une note

5.1.4 .Les concepts de groupement: représentés par les sous-systèmes, les paquetages.

Un paquetage : Le concept de paquetage unifié les concepts de catégorie et de sous système [Fannader &al, 00]. Un paquetage regroupe des éléments de la modélisation [Gabay, 98]: cas d'utilisation, classes, objets, modules ou composant. L'importation permet aux éléments d'un paquetage d'accéder aux éléments d'un autre package. Cette relation est à sens unique et est représentée par une relation de dépendance associée à un stéréotype « import »

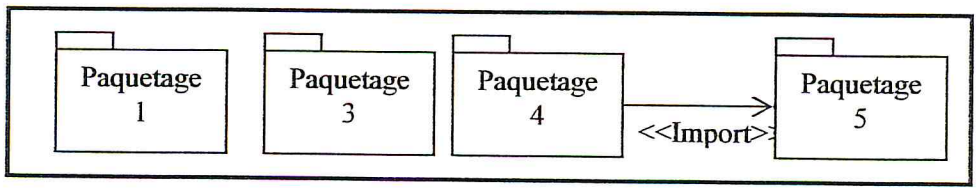


Figure II-20: Réprésentation graphique d'un paquatage

5.2 . Les relations

Elles permettent de relier les concepts entre eux. On distingue quatre types de relations, les associations, les généralisations, les agrégations, les compositions …etc. [Bernardi, 02].

5.2.1 .L'association: relation structurelle précisant que les objets d'un élément sont reliés aux objets d'un autre élément.

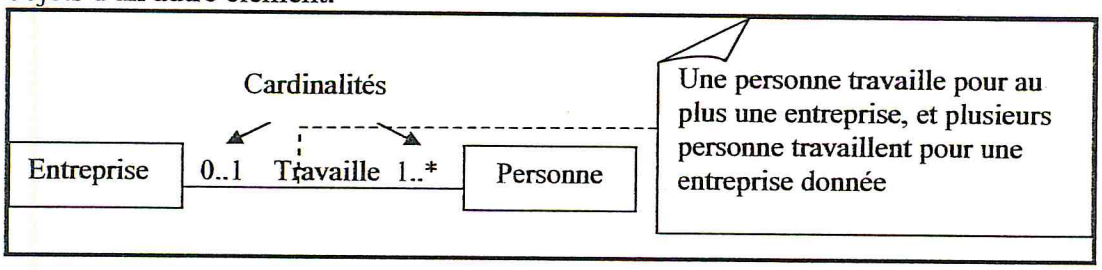


Figure II-21: Représentation graphique d'une association

5.2.2 .La généralisation: relation entre un élément général et un élément dérivé de celui-ci, mais plus spécifique (désigné par sous-élément ou élément fils).
Le plus souvent, la relation de généralisation est utilisée pour représenter une relation d'héritage.

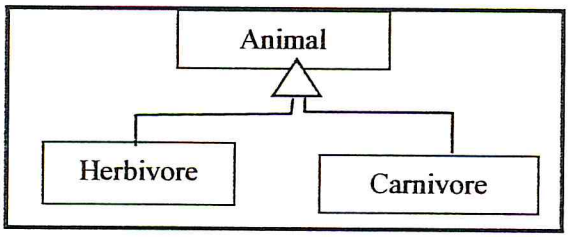


Figure II-22: Représentation graphique d'une généralisation

5.2.3 .L'agrégation : représente une association non symétrique dans laquelle une extrémité joue un rôle prédominant par rapport à l'autre extrémité [Muller, 97].

La figure suivante montre un exemple d'agrigation.

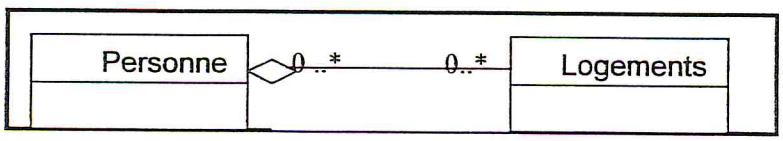


Figure II-23: Représentation graphique d'une agrégation

5.2.4 .La composition : relation d'agrégation mettant en avant une notion de propriété forte et de coïncidence des cycles de vie. Les parties sont créées et détruites en même temps que le tout [Bernardi, 02]. Un exemple de cette relation est représenté dans la Figure II-24.

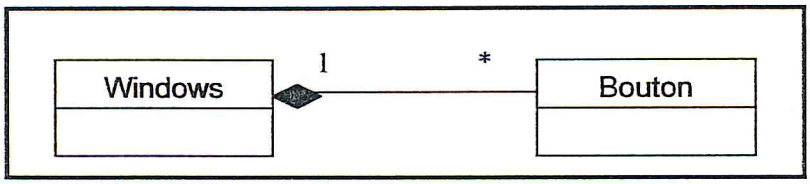


Figure II-24: Représentation graphique d'une composition

5.3 . Les diagrammes d'UML

UML définit neufs types de diagrammes qui sont présentés dans l'extrait du méta modèle suivant :

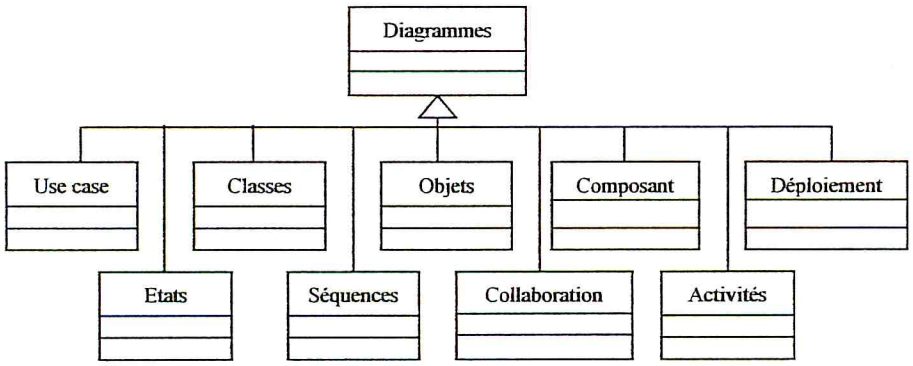


Figure II-25: Les diagrammes d'UML[Kettani& al, 01]

5.3.1 .Diagramme de cas d'utilisation

- **Un Acteur :** représente un rôle joué par une personne ou une chose qui interagit avec un système [Muller, 97]. Un acteur interagit avec les cas d'utilisations du système par des envois de messages.

Un diagramme de cas d'utilisation permet de décrire les interactions entre les acteurs de l'organisation et le système dans chacun des cas d'utilisation envisagés. Il décrit le comportement d'un système au point de vue de l'utilisateur et fixe les limites du système et les relations entre le système et l'environnement [Muller, 97].

Les divers cas d'utilisation du système vont être présentés dans les diagrammes de cas d'utilisation, comme nous montre la Figure II-26.

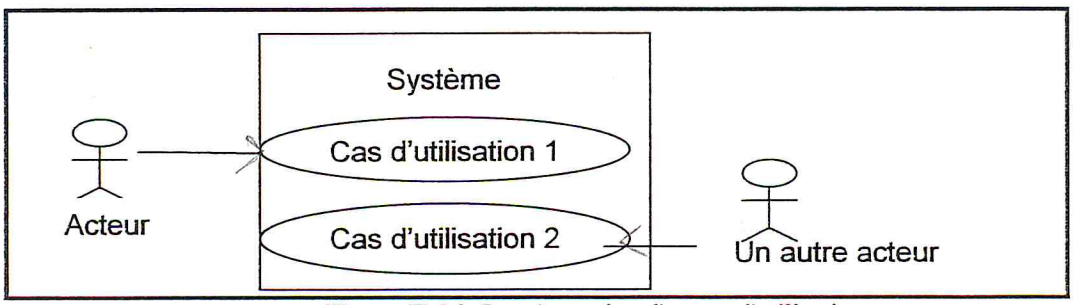


Figure II-26: Représentation d'un cas d'utilisation

Les cas d'utilisation peuvent être liés les uns aux autres par trois types de relations, illustrés dans la Figure II-27 :

➢ **La relation d'utilisation :** lorsque une ou plusieurs tâches sont utilisées régulièrement, on peut les factoriser dans un même use case et faire de telle sorte que d'autres use cases l'utilisent en le pointant par une flèche [Gabay 98].

Cet use case est en fait une sous partie de chaque use case qui l'utilise. Ce qui permet de décomposer un use case complexe en plusieurs uses cases.

➢ **La relation d'inclusion** : le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination. Cette relation a un caractère obligatoire (à la différence de la généralisation) et permet ainsi de décomposer des comportements partageables entre plusieurs cas d'utilisation différents [Gabay 98].

➢ **Le relation d'extension** : le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut-être soumise à condition. Cette relation permet de modéliser des variantes de comportement d'un cas d'utilisation [Gabay 98].

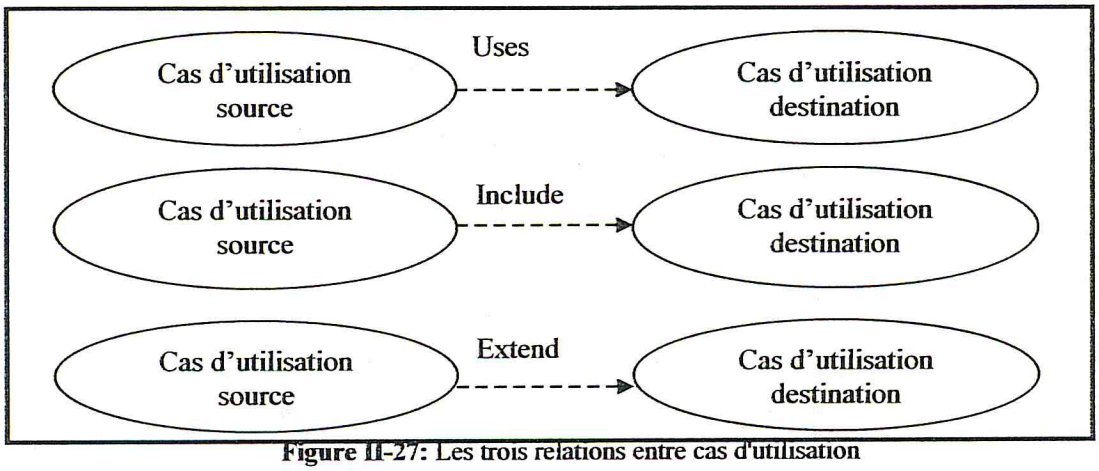


Figure II-27: Les trois relations entre cas d'utilisation

5.3.2 .Diagramme de classes

Les diagrammes de classes montre la structure du système et les éléments des classes tels que: les classes, les relations d'héritages entre classes, les associations, dont les agrégations, les attributs, les opérations et la spécification d'opérations et contraintes au niveau des entités [Bernardi, 02].

➢ **Stéréotype :** permet de définir une utilisation particulière d'éléments de modélisation existants ou de modifier la signification d'un élément [Bernardi, 02].

Exemple : les stréotypes de base d'UML sont :

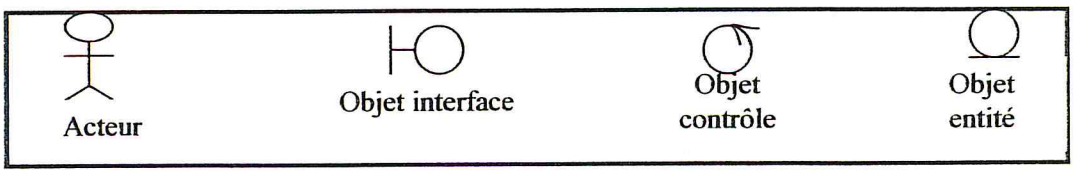


Figure II-28: Les stéréotypes d'UML

→*Acteur* : représente les rôles des interlocuteurs u système.

→*Objet interface* : représente les limites du système.

→*Objet contrôle* : représente les classes effectuant des traitements internes au système.

→*Objet entité* : représenté les objet de domaine [Fannader & al, 00].

5.3.3 .Diagramme d'objets

Ce type de diagramme UML montre des objets (instances de classes dans un état particulier) et des liens (relations sémantiques) entre ces objets [Fannader &al, 00].

Les diagrammes d'objets s'utilisent principalement pour montre un contexte, par exemple avant ou après une interaction, mais également pour faciliter la compréhension des structures de données complexes, comme les structures récursives [Muller, 97].

Un objet est représenté graphiquement comme montre la figure suivante :

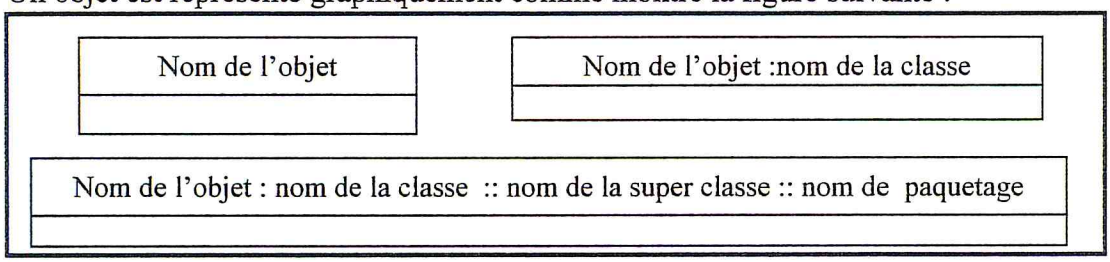


Figure II-29: Représetation graphique d'un objet

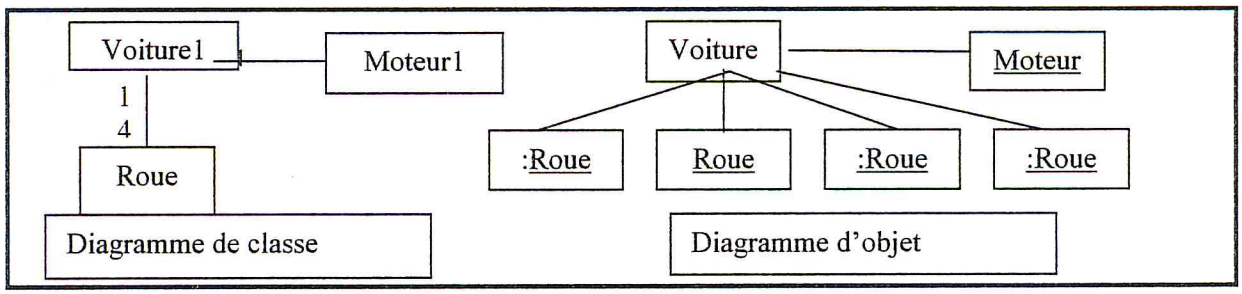


Figure II-30 : Exemple de diagramme d'objet et de classe

5.3.4 .Diagrammes de composants

➢ **Composant**: élément physique qui représente une partie implémentée d'un système. Il peut être du code, un script, un fichier de commandes, etc. les composants présentent un ensemble d'interfaces [Bernardi, 02].

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en terme de modules : fichiers sources, librairie, exécutables etc. [Booch, 00]. Ils décrivent les éléments physiques et leurs relations dans l'environnement de réalisation, ils montrent les choix de réalisation [Muller, 97].

Les composants du systèmes : le sous système, le module, le programme et le sous programme, le processus et la tâche.

➢ **Les sous systèmes** : la notion de sous système a déjà présentée au paragraphe consacré aux paquetages.

➢ **Module** : un sous système peut être décomposée en modules, chaque module correspond à un ensemble d'éléments physique (fichiers, bibliothèques, sous ensembles de logiciel.) [Gabay, 98]. La figure suivante donne le formalisme d'un module.

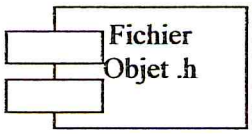


Figure II-31: Représentation d'un composant (fichier)

5.3.5 .Les diagrammes de déploiement

Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels. Ils peuvent monter des classes de noeuds ou des instances de noeuds [Muller, 97].

Un diagramme de déploiement permet également de représenter les relations entre différents noeuds.

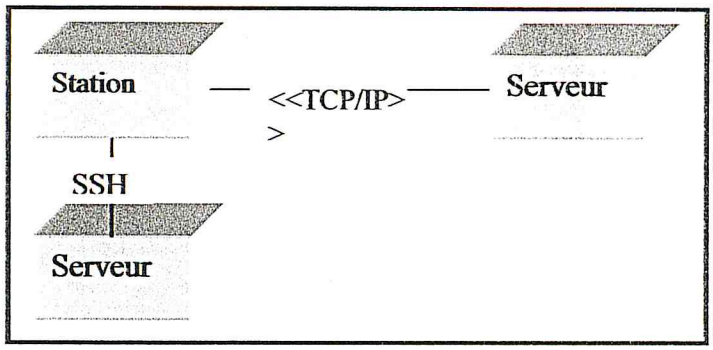


Figure II-32: Représentation de relations entre nœuds

5.3.6 .Diagramme de séquences

Les diagrammes de séquences permettent de représenter les interactions entre objets en précisant la chronologie des échanges de messages. Ils peuvent être utilisés pour représenter les scénarios d'un cas d'utilisation donnée [Gabay, 98].

➢ **Interactions**: modélisent un comportement dynamique entre objets [Muller, 97]. Elles se traduisent par l'envoi de messages entre objets. Un diagramme de séquence représente une interaction entre objets, en insistant sur la chronologie des envois de messages [Bernardi, 02].

➢ **Les messages :** Les messages échangés sont représentés au moyen de flèches horizontales partant de l'émetteur vers le récepteur. L'ordre de l'envoi est donné par la positions sur l'axe vertical [Fannader, 00].

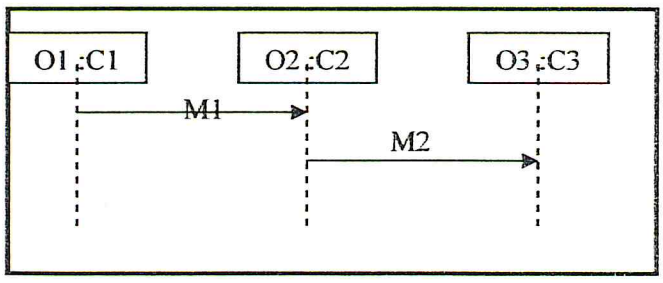


Figure II-33: Agencement de messages

Le diagramme de séquence distingue 5 types de messages prédéfinis qui sont présentés dans le tableau suivant :

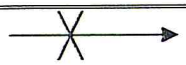
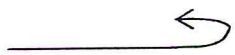
Type de message	La signification
	Message simple, exemple : à une passation de contrôle en mono tâche.
	Message asynchrone, pas de réponse attendue par l'émetteur.
	Message synchrone, réponse nécessaire du destinataire
	Message dérobant, le destinataire doit être à l'écoute
	Message minuté, l'émetteur est bloque pendant un laps de temps.

Tableau II-1 : Les différents types de messages

➢ **Période d'activation**

Correspond au temps pendant lequel un objet effectue une action, soit directement, soit par l'intermédiaire d'un autre objet.

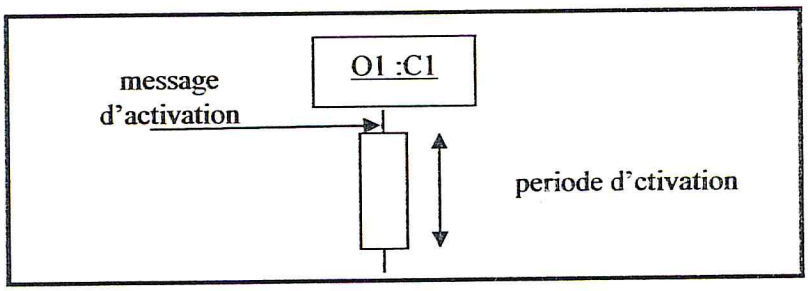


Figure II-34: Activation d'un objet de manière simple

5.3.7 .Diagrammes de collaboration

Les Diagrammes de collaboration montrent des interactions entre les objets et les acteurs. Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent et peuvent être utilisé pour identifier les principaux objets [Gabay, 98]. La figure suivante présente le formalisme de base d'un diagramme de collaboration : un échange de message entre deux objets.

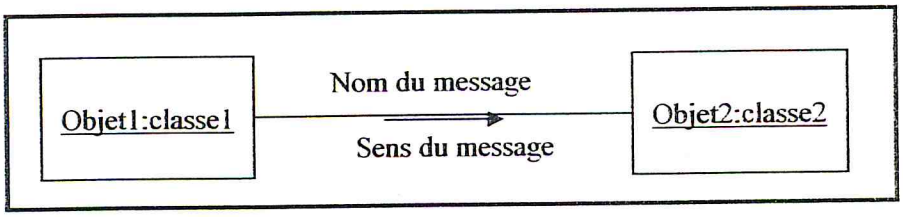


Figure II-35: Formalisme de base du diagramme de collaboration

5.3.8 .Diagramme d'état-transition

Un diagramme d'état-transition est un graphe constitué de nœuds représentants des états ainsi que des flèches représentant des transitions, portants des paramètres et des noms d'événements [Fannader &al, 00]. Les diagrammes d'états permettent de définir le comportement d'un objet particulier vis-à-vis des sollicitations internes ou externes auxquelles il peut être soumis [Gabay, 98]. Ils permettent aussi de décrire l'évolution dans le temps les états des objet d'une certaine classe, les événements auxquels ils réagissent et les transitions qu'ils effectuent.

Les diagrammes d'états visualisent des automates (Figure suivante) d'états finis, du point de vue des états et des transitions [Muller, 97].

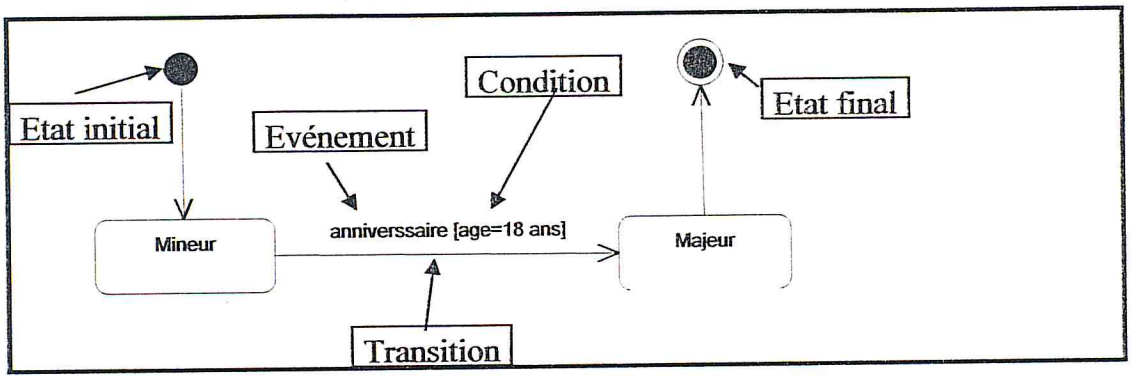


Figure II-36: Exemple de diagramme d'état-transition

➢ **Points d'exécution des opérations**

Il existe six points pour spécifier les opérations qui doivent être exécutées. Ces points sont dans l'ordre d'exécution :

- L'action associé à la transition d'entrée (op1).
- L'action d'entrée d'état (op2).
- L'activité dans l'état (op3).
- L'action de sorite d'état (op4).
- L'action associée aux événements internes (op5).
- L'action associée à la transition de sortie de l'état (op6).

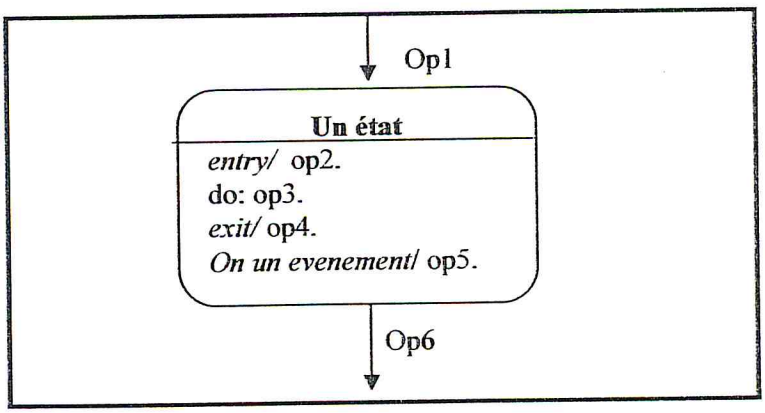


Figure II-37: Les points d'exécution pour un état [Muller 97]

5.3.9 .Diagramme des activités

Ce diagramme permet de décrire le déroulement d'un cas d'utilisation particulier. Il est possible de décrire les acteurs responsables de chaque activités par l'utilisation des «couloirs d'activités» qui permettent de répartir graphiquement les différentes activités entre les acteurs opérationnels [Muller, 01]. Chaque activité est placée dans le «couloir» correspondant à l'acteur qui assume cette activité.

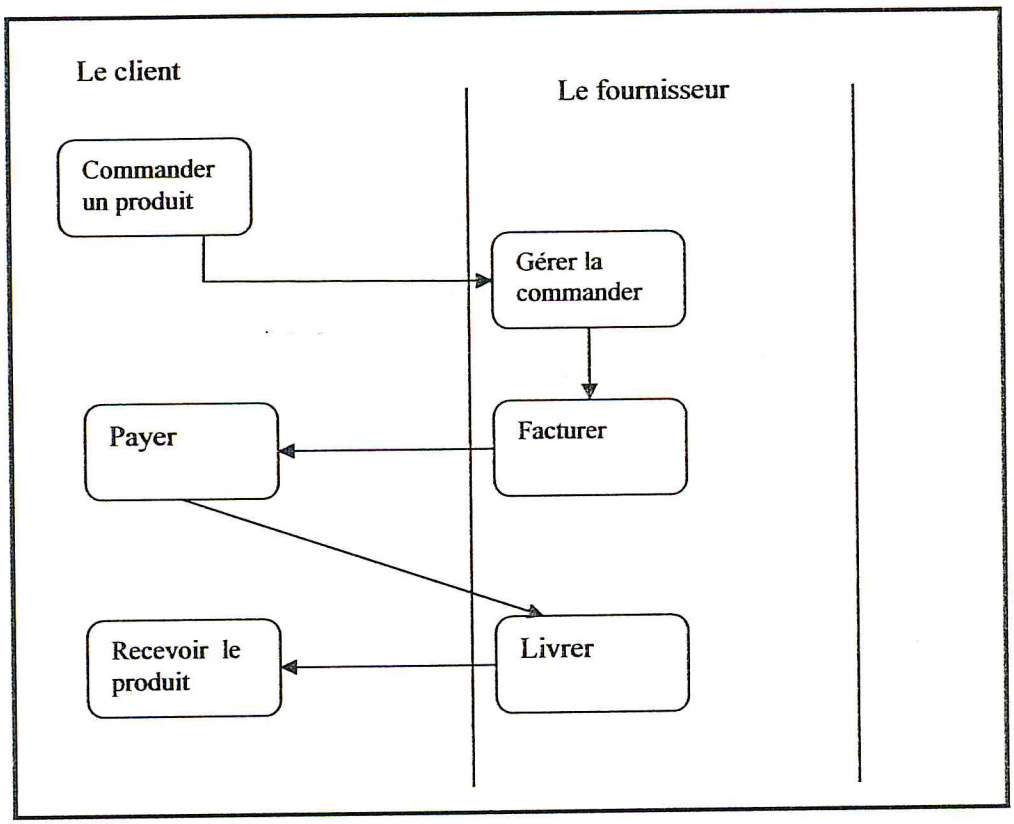


Figure II-38: Exemple d'un diagramme d'activité

6. Conclusion

Dans ce chapitre nous avons présenté deux standards de modélisation et de spécification de données, UML et EXPRESS, soulignant leurs manières de la représentation de données.

UML est un langage ou plutôt un formalisme de modélisation orienté objet, il permet de représenter les systèmes entiers par des concepts objets. UML est un langage orienté objet basé sur une notation graphique.

EXPRESS peut être utilisé pour la spécification de données, où les données sont soumises à une batterie de contraintes. L'inconvénient majeur d'EXPRESS réside dans son incapacité à spécifier les comportements des systèmes, ce qui est possible par UML via les fameux diagrammes, cas d'utilisation, état-transition, ...etc. Dans le chapitre suivant nous utiliserons ces deux langages pour le développement de l'outil de définition de workflow.

CHAPITRE III.
LA DEMARCHE DE DEVELOPPEMENT DE L'OUTIL

1. Introduction

Si un système de gestion de workflow est implémenté intelligemment et sur la base d'un sérieux travail de modélisation des processus, il peut assurer de grands services et des réductions de coût aux organisations. Ce système regroupe les composantes logicielles qui interprètent et stockent les définitions de processus. Dans ce présent travail nous développons un logiciel qui représente un outil de définition de workflow.

Le développement de l'outil est une succession d'étapes par laquelle passe un logiciel, ce développement est difficile à réussir. Afin d'augmenter leurs chances de réussite, des méthodes de développement de logiciel ont été définies, ces méthodes permettent de mieux organiser, d'avoir une meilleure compréhension, de réduire la complexité des applications, et permettent une plus grande facilité dans l'interprétation des concepts logiciels.

Nous commençons ce chapitre par le choix de la méthode de développement suivie, et pour réaliser notre outil nous présentons toutes les étapes de développement de l'outil qui sont:

La spécification des besoins, l'analyse, la conception, l'implémentation, le test et la validation.

2. Choix de la démarche suivie

Une méthode de développement de logiciel est définie pour représenter le processus, elle comprend [Muller, 97] :

- Des éléments de modélisations qui sont les briques conceptuelles de base.
- Une notation dont l'objectif est d'assurer le rendu visuel des éléments de modélisation.
- Un processus qui décrit les étapes à suivre lors du développement du système.

La notation unifiée UML que nous avons utilisées est basée sur les méthodes d'analyse et de conception : BOOCH, OMT et OOSE (Use/Cases), de ce fait, elle permet de couvrir le cycle de vie d'un logiciel depuis l'analyse du besoin jusqu'au test. Cette notation est d'une très grande richesse. Elle permet de couvrir toutes les phases du développement :

– Les besoins des utilisateurs du futur système, exprimés à l'aide de cas d'utilisation

– La spécification complète du système, sous forme de diagrammes couvrant les parties statiques et dynamiques du système.

– La conception détaillée, jusqu'à un niveau très proche du code en langage objet de l'implémentation.

–Les suites de tests permettant de s'assurer qu'une implémentation candidate est effectivement conforme à la spécification élaborée en phase amont, sous forme de diagrammes de séquences.

La notation UML est un langage de modélisation et non pas une méthode objet [Muller, 97], c'est à dire UML ne décrit pas une démarche de développement de logiciel.

Le processus de développement que nous avons suivi est le cycle de vie en cascade, ce modèle décrit par Royce en 1970 a été largement employé depuis, pour la description générale des activités liées aux logiciels [Muller, 97]. Il décrit le cycle de vie d'un logiciel par une suite de phases qui s'enchaînent dans un déroulement linéaire (Figure III-1), depuis l'analyse des besoins jusqu'à la maintenance [Muller, 97].

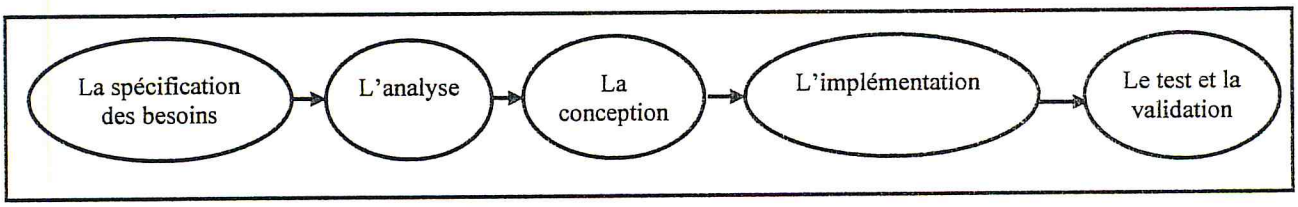


Figure III-1: Cycle de vie d'un logiciel [Brès, 93]

3. La spécification des besoins

La finalité de cette étape est la description générale des fonctionnalités du système. En répondant à ces questions : "quelles sont les fonctions du système ?", "quels sont les utilisateurs du système?", "et qu'attendent-ils du système ?". Cette étape étudie le comportement du système exprimé sous la forme des cas d'utilisation, le contexte du système, les acteurs et les scénarios.

3.1 . Lescas d'utilisation

La spécification détermine le quoi faire, il s'agit de définir les besoins de l'utilisateur. L'expérience montre que la technique des cas d'utilisation (use cases) se prête bien à la détermination des besoins d'utilisateurs [Muller, 97].

L'étude des cas d'utilisation débute par la détermination des acteurs (catégories d'utilisateurs) du système.

3.1.1 .Définition des acteurs

Notre système est destiné à des utilisateurs, c'est pour eux que le système est fabriqué. Ils sont la réponse à la question : pour qui est fabriqué le système ?

Les acteurs de notre système sont:

- **Un expert de métier:** est un utilisateur qui est chargé d'effectuer la définition du processus, il est donc l'acteur principal du système.

- **Un opérateur:** est un acteur secondaire, il ne manipule pas notre système, il peut seulement consulter les processus.

3.1.2 .Les cas d'utilisation

Les différentes fonctionnalités offertes par notre outil forment un ensemble de *cas d'utilisation* (*"Use Case"*). UML propose au travers les diagrammes des cas d'utilisation de répertorier et de structurer les fonctionnalités que le futur système offrira à ses utilisateurs. Les différents cas d'utilisation recensés pour le système sont présentés dans la section suivante :

3.1.2.1 . Définition de processus

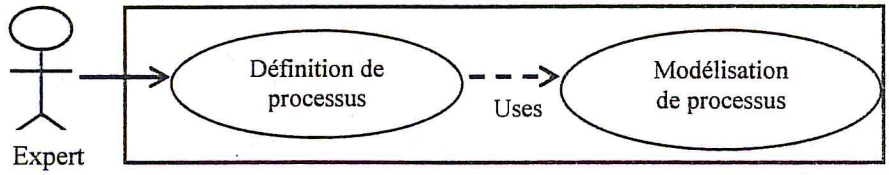


Figure III-2: Cas d'utilisation « définition des processus »

Le cas « définition des processus » utilise un autre cas qui est « la modélisation des processus » ceci est représenté par la relation <<uses>>.

3.1.2.2 . Modélisation de processus

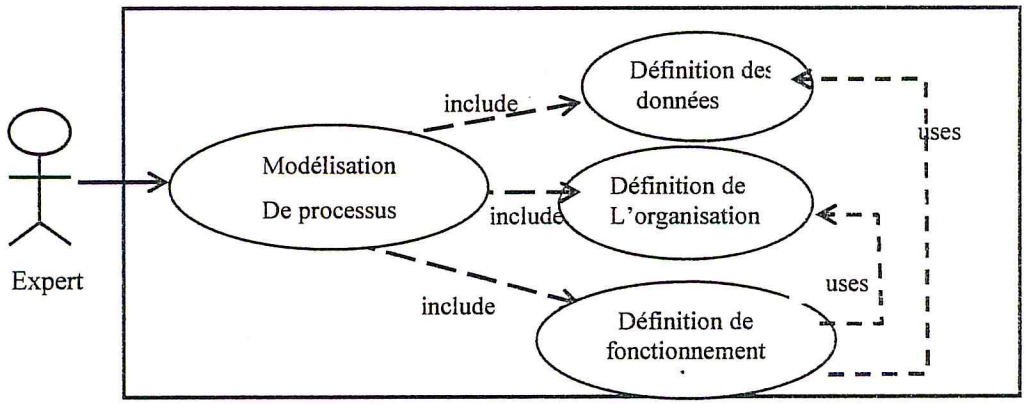


Figure III-3: Cas d'utilisation « modélisation des processus »

Pour modéliser un processus, il faut définir au début l'organisation et les données indépendamment du processus, puis introduire le fonctionnement du processus, ceci explique que le cas d'utilisation « modélisation des processus » inclut les trois cas : définition de processus, de l'organisation et du SI, cette inclusion est exprimé par la relation <<include>>.

3.1.2.3 . Définition de l'organisation, des données et de fonctionnement du processus

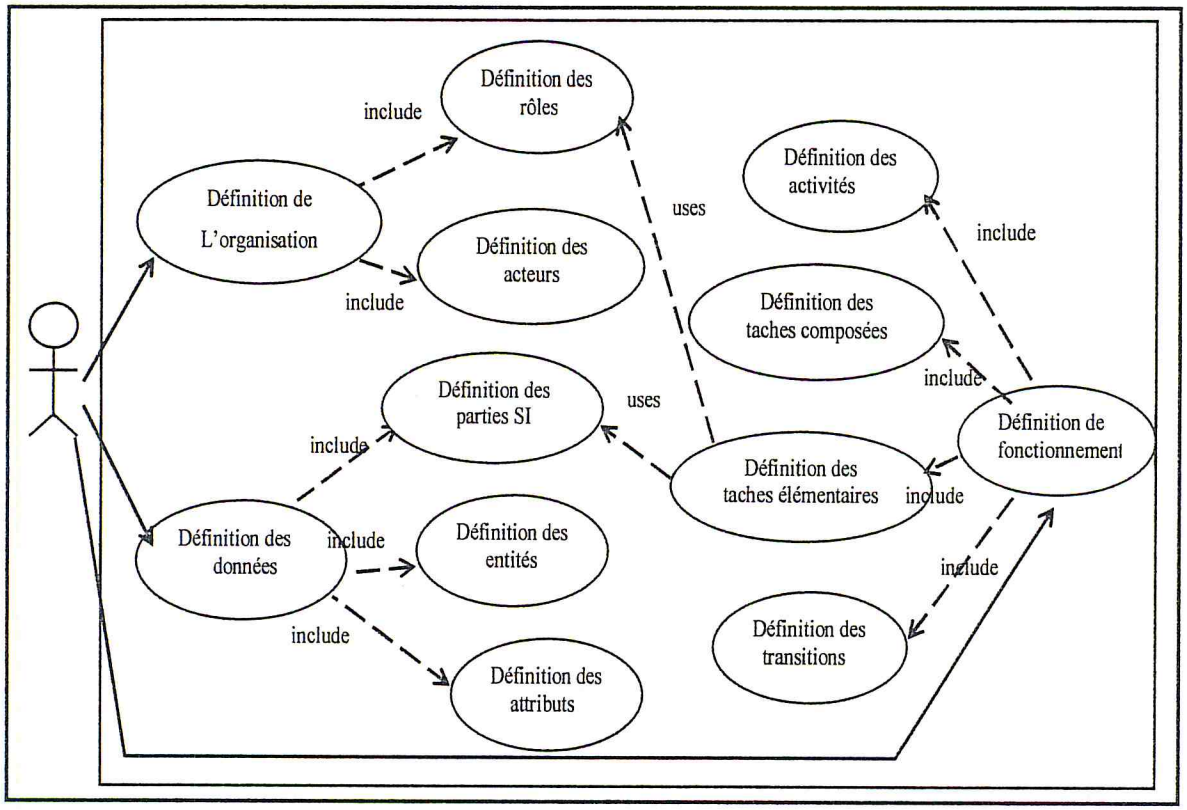


Figure III-4: Cas d'utilisation «définition de fonctionnement, des données et d'organisation »

Le cas d'utilisation «définition de fonctionnement » utilise les deux cas : définition de l'organisation et des données, en effet le cas « définition des taches élémentaires utilise « la définition des rôles » et « la définition des parties SI ».

3.1.2.4 . La consultation des processus

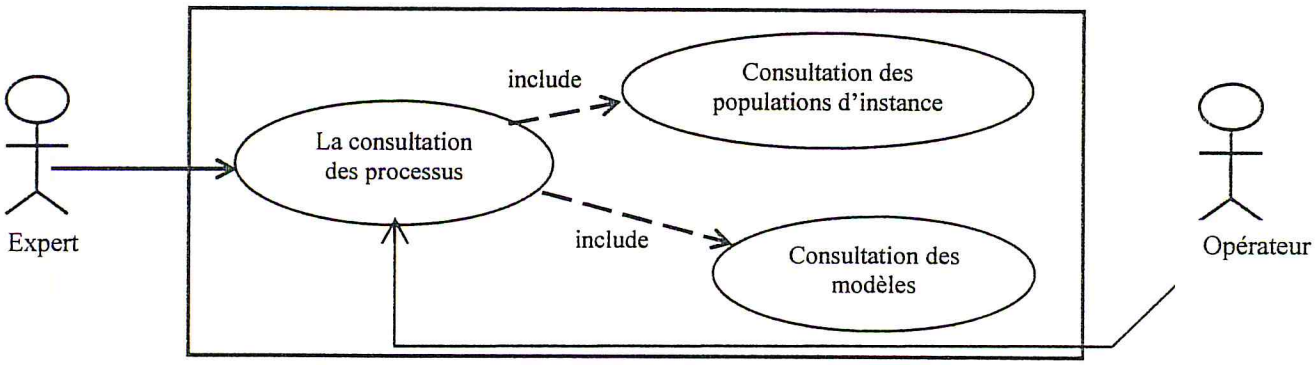


Figure III-5: Cas d'utilisation «consultation des processus »

- Le cas « consultation des processus » est réalisé par la consultation des instances et celle des modèles, il est présenté par la relation <<include>>.

3.1.2.5 . La mise à jour des processus

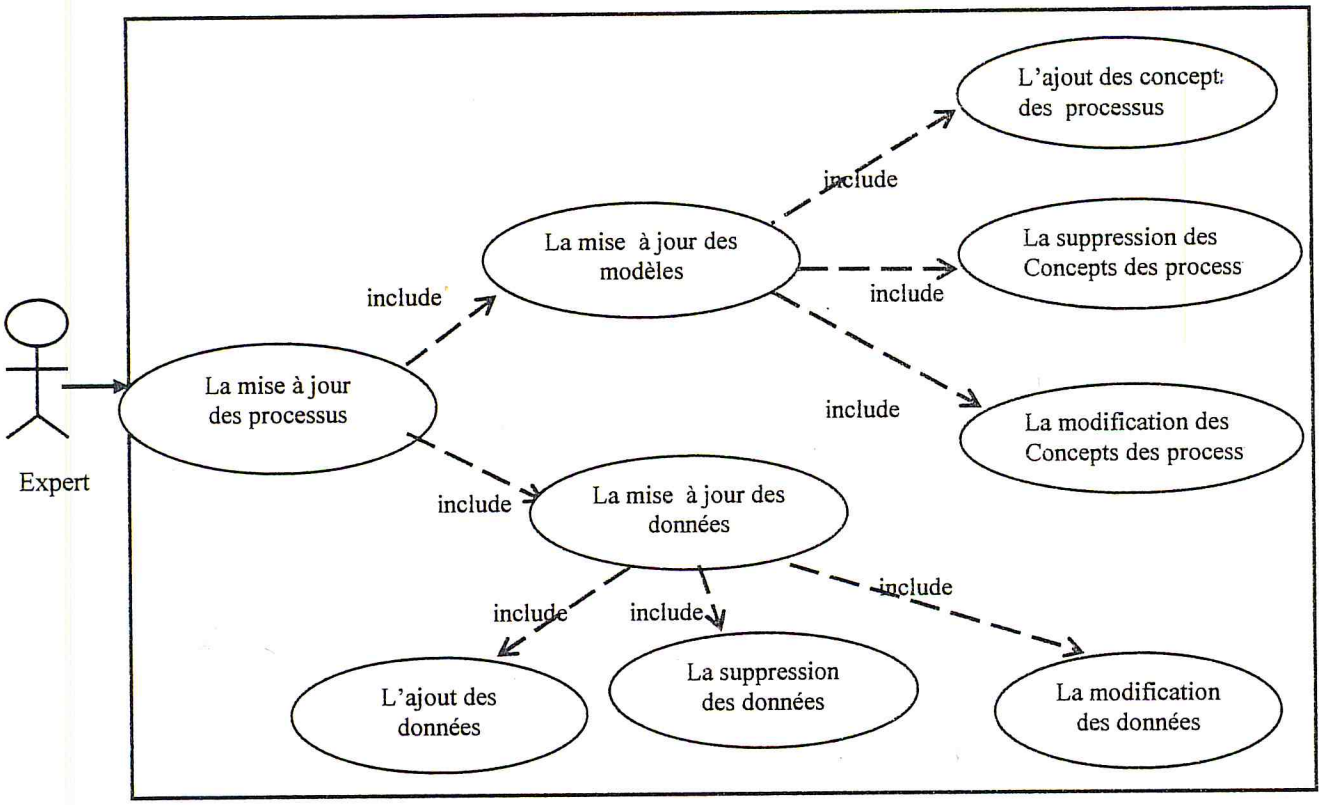


Figure III-6: Cas d'utilisation « la mise à jour des processus »

- La mise à jour des processus inclut deux cas, la mise à jour des modèles et celle de données,
- La mise à jour des modèles inclut trois autre cas : l'ajout, la modification et la suppression des concepts des processus.
- La mise à jour des données comprend l'ajout, la modification et la suppression des données.

La figure suivante illustre l'ensemble des fonctionnalités offertes par notre système :

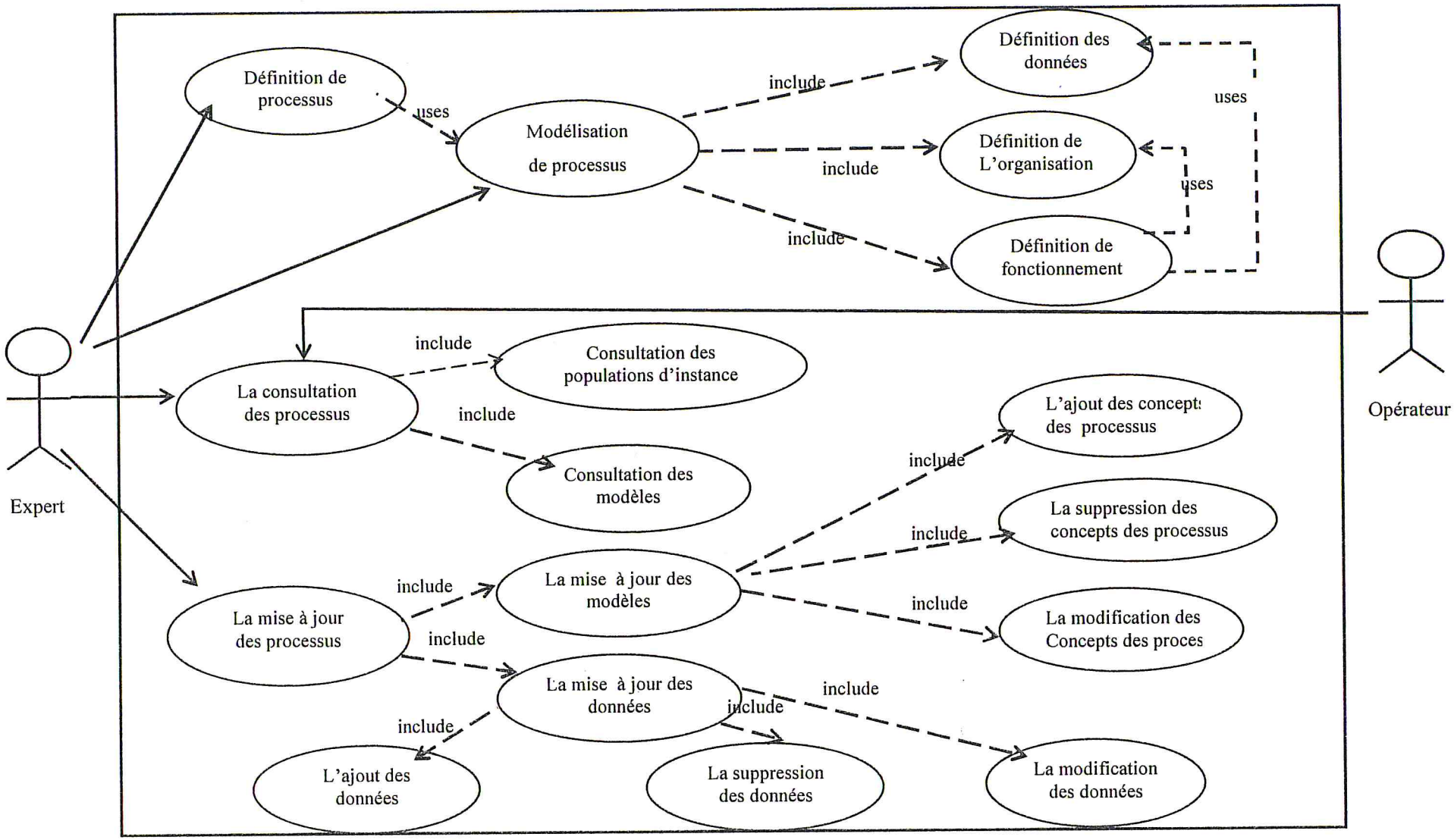


Figure III-7: Le diagramme de cas d'utilisation de ODWF

3.1.3 .Diagramme de séquence

Les cas d'utilisation de UML ont certes l'avantage d'être graphiquement très simples et donc faciles à appréhender. Malheureusement, cette simplicité ne va pas sans une certaine pauvreté sémantique [Muller, 97], cependant les diagrammes de séquence permettent de bien schématiser les scénarios des cas d'utilisation et montrent les interactions entre plusieurs objets.

3.1.3.1 . Définition de processus

Le scénario :

→ Après la modélisation de processus, l'expert demande au système d'enregistrer ce processus.

→ Le système envoie une commande de génération de population d'instances à Ecco (Ecco est un contrôleur de modèles EXPRESS, nous détaillons cet outil dans l'annexe C)

→ Ecco envoie une réponse au système après une vérification.

→ S'il y a un problème au niveau de la base de données, un message d'erreur est envoyé à l'expert.

Le diagramme de séquence suivant exprime le cas d'utilisation « définition de processus ».

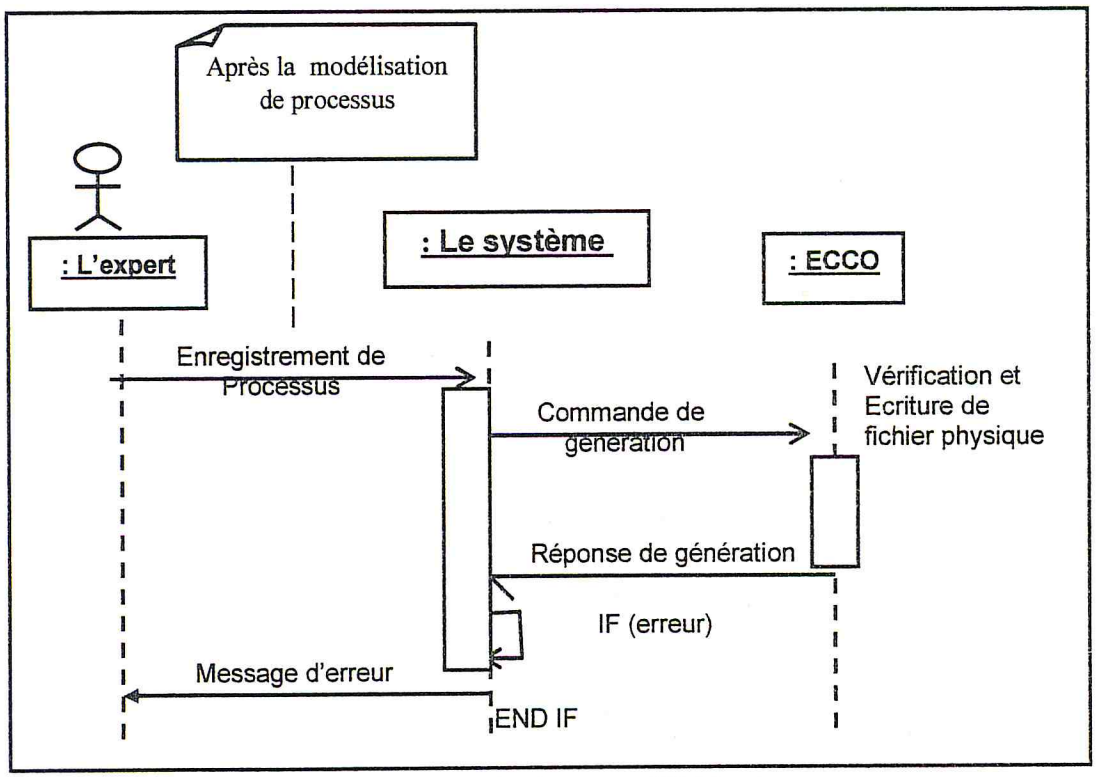


Figure III-8: Le diagramme de séquence « définition de processus »

3.1.3.2 . Modélisation de processus

1) Définition de l'organisation

La définition de l'organisation consiste à définir les rôles qui participent à l'accomplissement des tâches, ainsi que ses acteurs.

Le scénario: « définition des acteurs»

→ L'expert ajoute un nouvel acteur.

→ Le système lui affiche les propriétés de l'acteur.

→ L'expert remplit les propriétés de l'acteur

Le diagramme de séquence suivant exprime le cas d'utilisation « définitions des acteurs ».

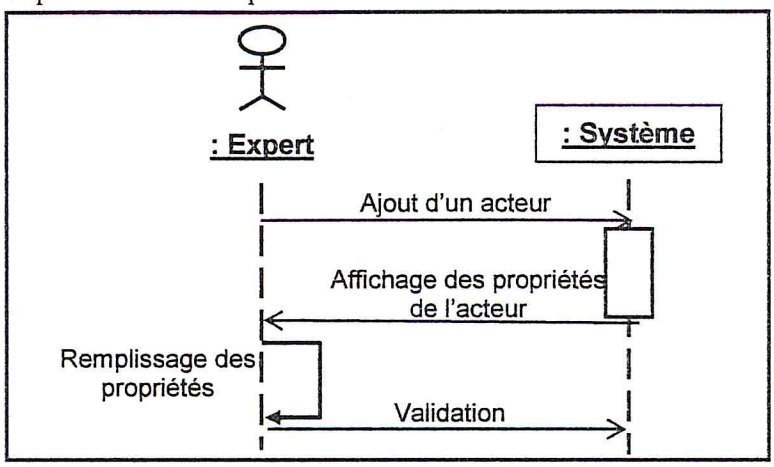


Figure III-9: Le diagramme de séquence « définition des acteurs »

Le scénario : « définition des rôles»

→ L'expert ajoute un nouvel rôle.

→ Le système lui affiche les propriétés du rôle.

→ L'expert remplit les propriétés du rôle.

Le diagramme de séquence suivant exprime le cas d'utilisation « définition des rôles ».

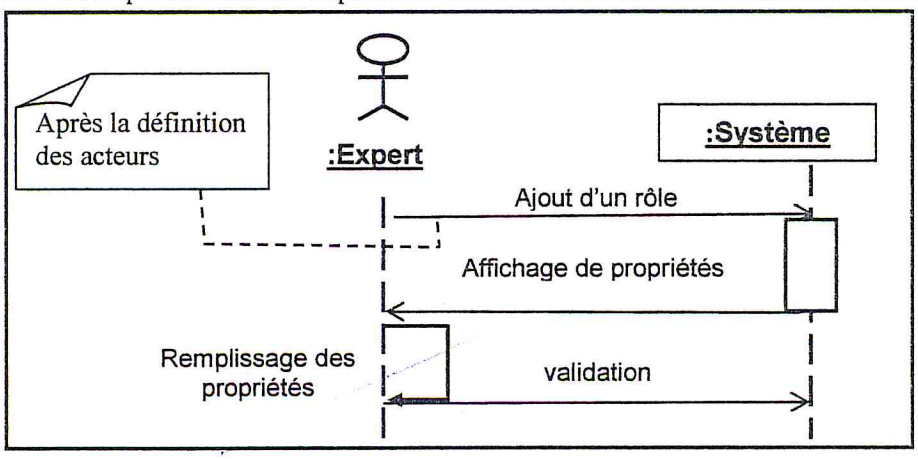


Figure III-10: Le diagramme de séquence « définition des rôles »

2) Définition des données

Le scénario : « définition des parties SI »

→ L'expert ajoute une nouvelle partie SI.

→ Le système lui affiche les propriétés de cette partie.

→ L'expert remplit les propriétés de la partie SI, ainsi que les entités qui composent cette partie.

Le diagramme de séquence suivant exprime le cas d'utilisation « définition des parties SI » :

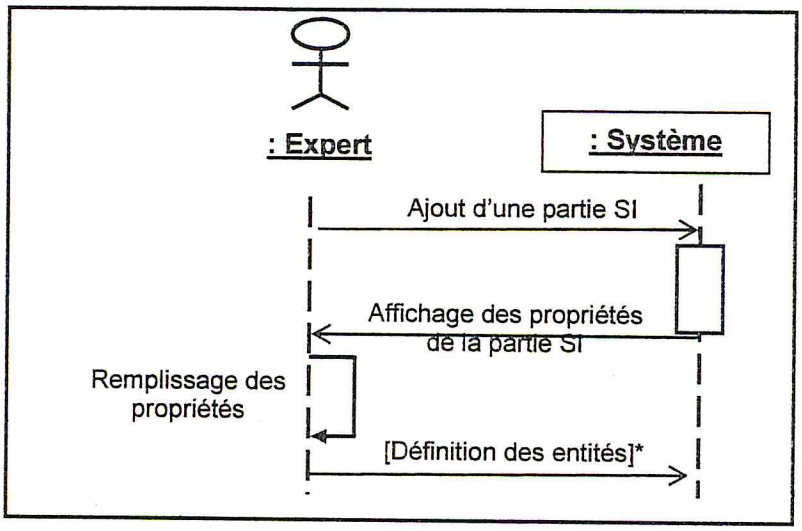


Figure III-11: Le diagramme de séquence « définition d'une partie SI »

3) Définition de fonctionnement

Le scénario : « Définition d'une tâche»

→ L'expert ajoute une nouvelle tâche.

→ L'expert place la tâche dans le panneau de modélisation.

→ Le système lui affiche les propriétés de cette tâche.

→ L'expert remplit les propriétés de la tâche.

Le diagramme de séquence suivant exprime le cas d'utilisation « définition d'une tâche ».

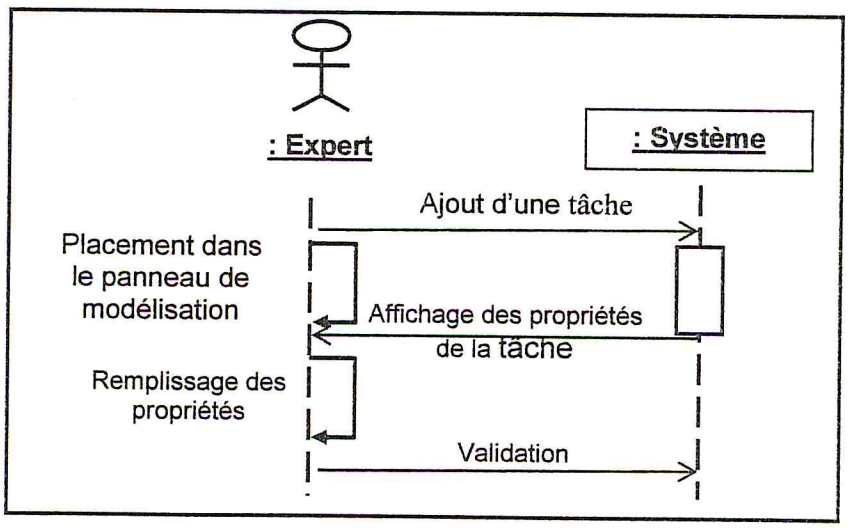


Figure III-12: Le diagramme de séquence « définition d'une tâche »

Le scénario : « modification des données »

→ L'expert choisit le processus qu'il veut modifier.

→ Le système recherche et affiche ce processus.

→ L'expert choisit l'objet à modifier.

→ Le système lui affiche les propriétés de cet objet.

→ L'expert modifie les propriétés de l'objet.

→ Le système envoie une commande de modification à Ecco pour enregistrer les données modifiées dans la base de données.

→ Ecco envoie une réponse d'enregistrement dans la base de données au système après une vérification s'il y a une erreur, un message d'erreur est envoyé à l'expert via le système.

Le diagramme de séquence suivant exprime le cas d'utilisation « modification des données ».

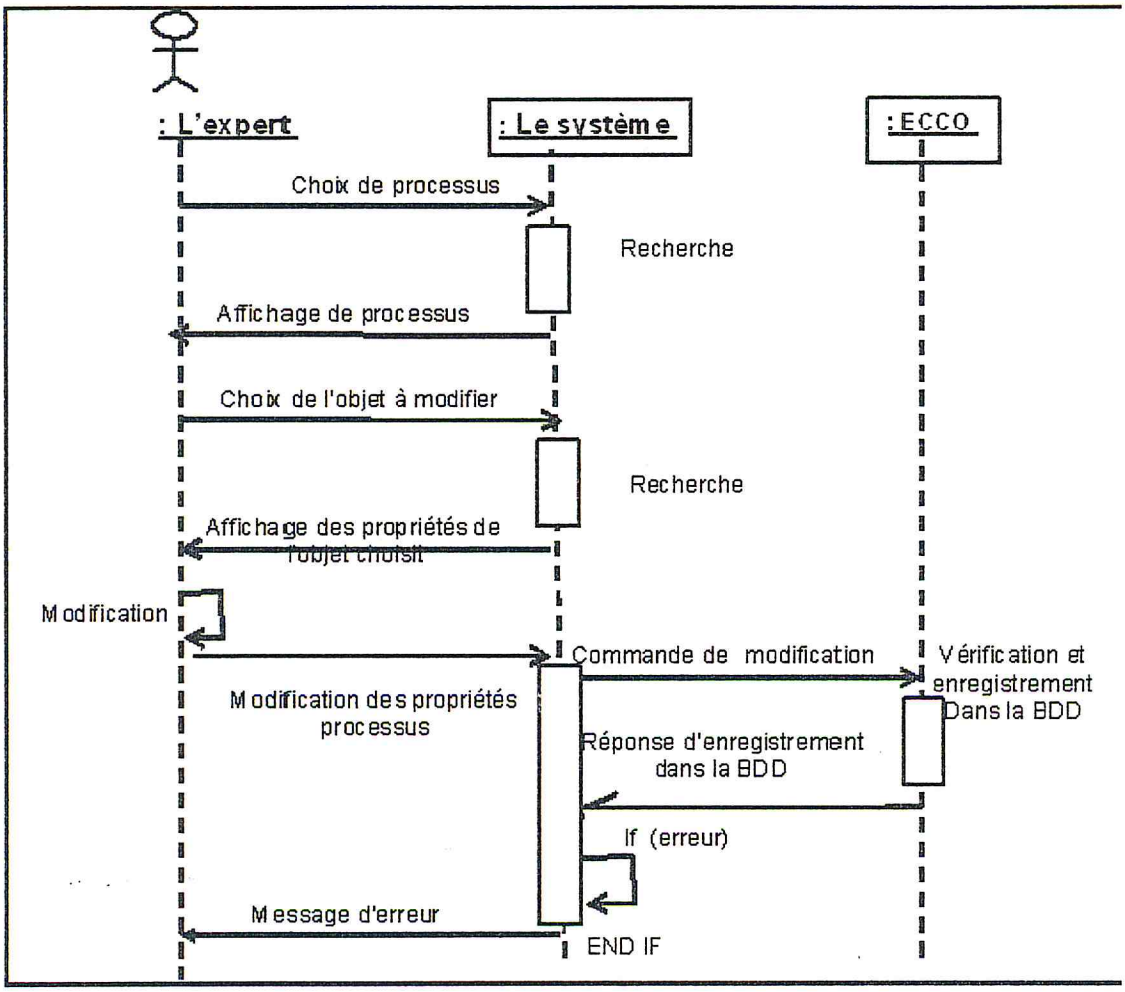


Figure III-13: Le diagramme de séquence « modification des données »

3.1.4 .Diagramme de collaboration

Les fonctionnalités décrites par les cas d'utilisation sont réalisées par des collaborations d'objets du domaine [Muller, 97], d'où il est envisageable, d'employer les diagrammes de collaborations bien que ce dernier ne soit qu'une variante des diagrammes de séquences et exprime de ce fait la même sémantique.

Nous avons choisi comme un exemple: définition de processus

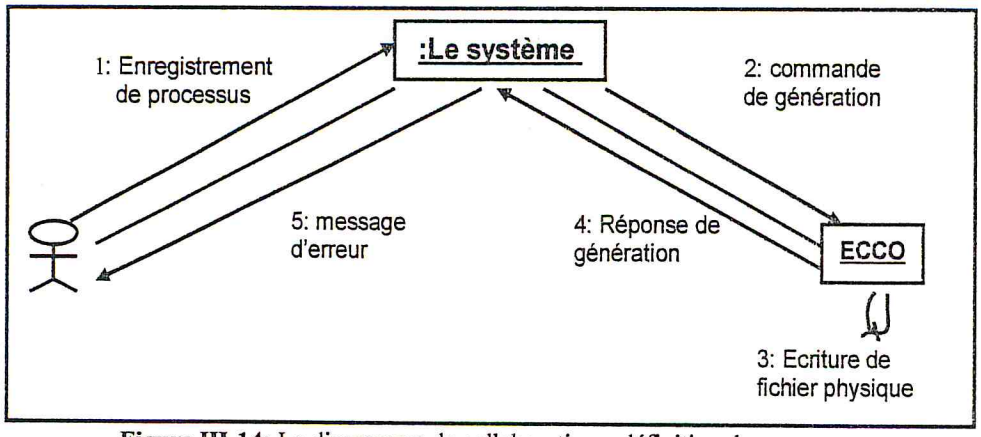


Figure III-14: Le diagramme de collaboration « définition de processus »

3.1.5 .Le diagramme d'activité

Les diagrammes d'activités sont utilisés pour formaliser les comportements des cas d'utilisation.

Le diagramme d'activité suivant décrit le déroulement du cas d'utilisation « modification des données ». (ODWF : est notre Outil de Définition de WorkFlow).

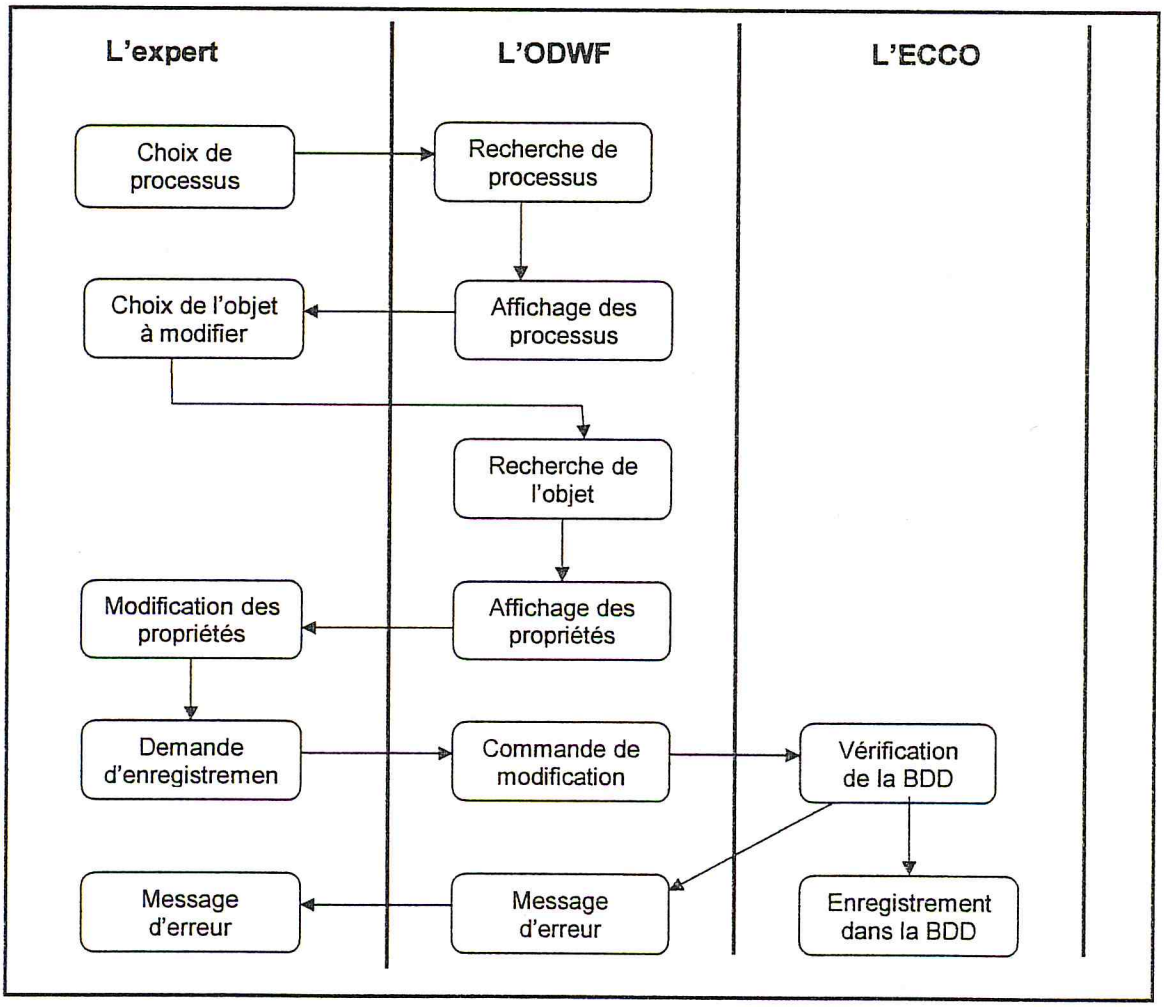


Figure III-15: Le diagramme d'activité « modification des données »

Le diagramme d'activité suivant décrit le déroulement du cas d'utilisation « définition de processus ».

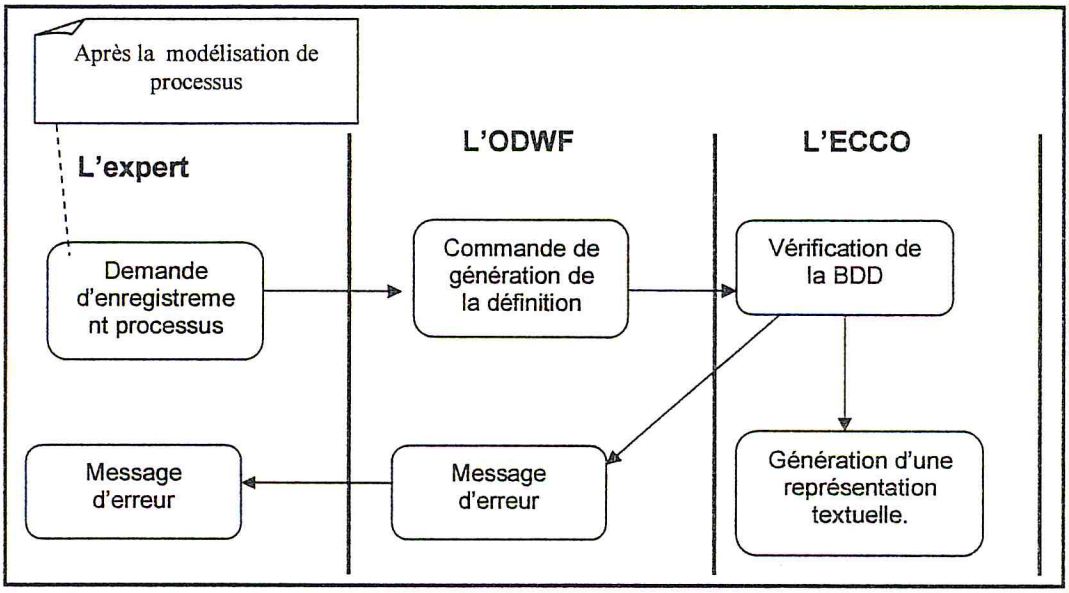


Figure III-16: Le diagramme d'activité de « définition de processus »

4. L'analyse

L'étape d'analyse doit fournir une approche conceptuelle du problème [Sommerville, 88].

Le but de cette étape est de définir une structure robuste et extensible qui nous servira de base pour la construction du système [Brès, 93].

L'objectif de cette phase est de déterminer les éléments intervenant dans le système à construire, ainsi que leur structure et leurs relations.

Pour réaliser la modélisation, il faut avoir un méta modèle regroupe l'ensemble des concepts d'un processus workflow, l'instanciation de ce méta modèle nous donne un modèle de processus.

Après l'étude des différents concepts d'un processus workflow dans le chapitre I, un processus est vu comme étant un ensemble d'activités, chaque activité est un ensemble de tâches partiellement ordonnées, et chaque tâche est accomplie par un rôle approprié, qui peut être joué par un ou plusieurs acteurs, s'il s'agit respectivement d'un rôle individuel ou d'un rôle de groupe. Un acteur qu'il soit humain ou outil peut jouer plusieurs rôles dans un processus.

4.1 . Les différentes vues d'un processus

Notre analyse de l'univers de workflow nous a permis de dégager les concepts présentés dans la section précédente. Ces concepts peuvent être classés dans 3 principales catégories suivant 3 différentes vues (

Figure III-17) : fonctionnelle, organisationnelle et informationnelle.

→ **La vue fonctionnelle** : met l'accent sur la modélisation du processus en hiérarchie : processus, activité, tâche…etc.

→ **La vue organisationnelle** : modélise l'organisation d'une entreprise, représentée par l'acteur ainsi que le rôle.

→ **La vue informationnelle** : définit l'ensemble de données manipulées par une tâche.

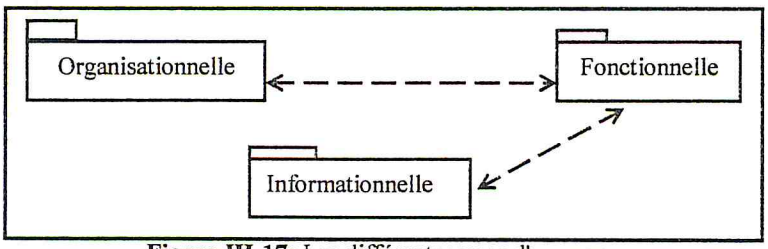
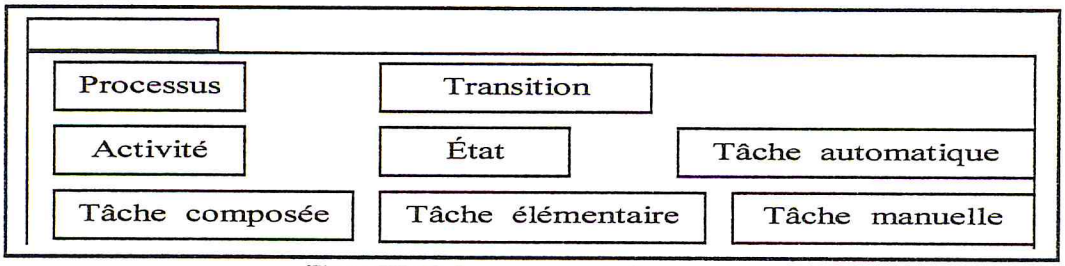


Figure III-17: Les différentes vues d'un processus

4.1.1 .Vue fonctionnelle

Elle permet de représenter les processus d'une organisation, elle s'intéresse à la description d'un processus en terme de ses constituants, indépendamment des moyens et des ressources mis en œuvre pour les réaliser. Elle offre plusieurs niveaux de visualisation en permettant de décomposer un processus en activités et l'activité en tâche.

Le paquetage suivant illustre les composants d'un processus :



FigureIII-18: Les composants d'un processus

Le diagramme de classe suivant présente les différents composant de cette vue d'une manière détaillée :

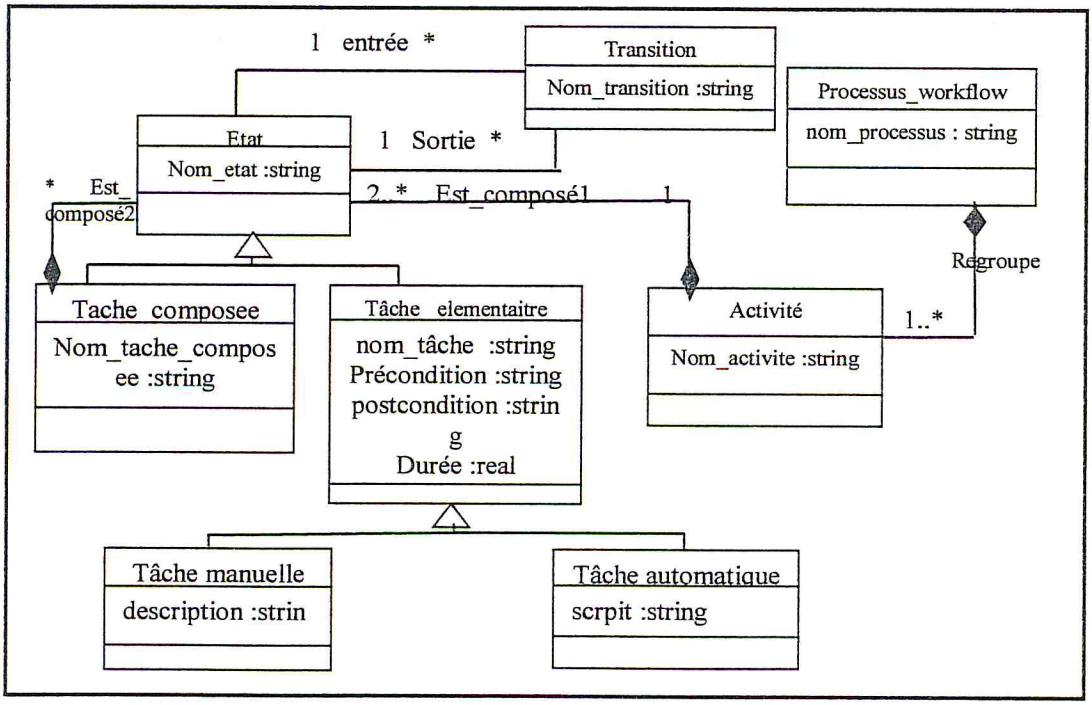


Figure III-19: Le diagramme de classe de la vue : processus

4.1.2 .Vue organisationnelle

Cette vue concerne les moyens humains, techniques et organisationnels utilisés pour accomplir les objectifs détaillés sous la forme d'activités, elle est réalisée de manière collaborative par un ensemble d'acteurs auxquels sont assignés des *rôles*. Un *acteur* exerce plusieurs rôles, un rôle est attribué à plusieurs acteurs.

Le paquetage suivant illustre les composants d'une organisation:

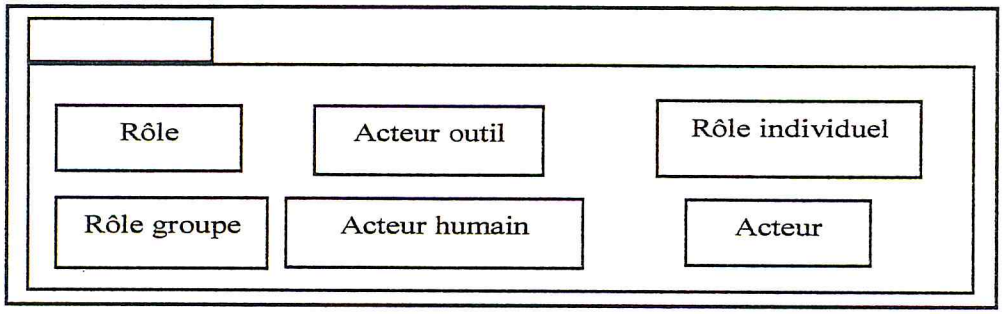


Figure III-20 : La vue des composants de l'organisation

Le diagramme de classe suivant présente les différents composants de cette vue d'une manière détaillée :

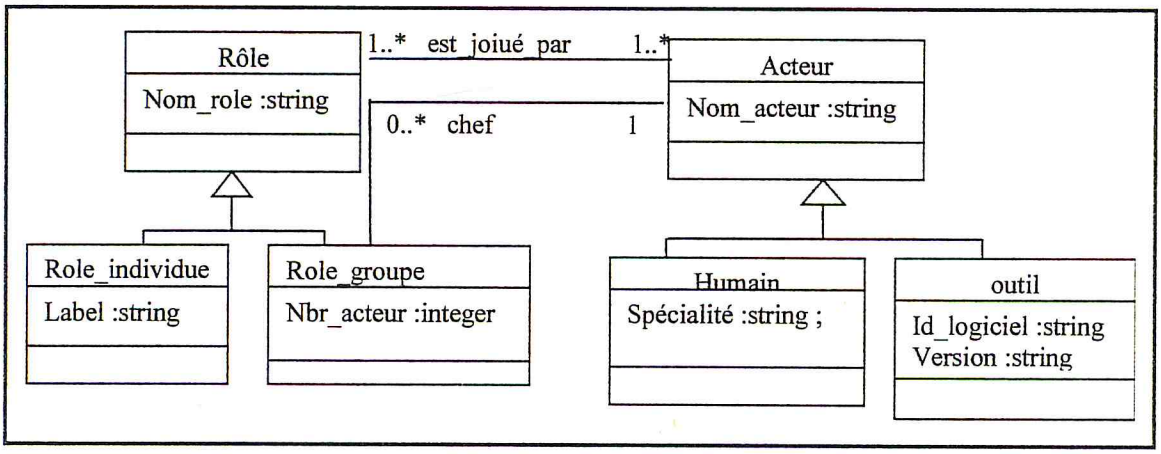


Figure III-21: Le diagramme de classe « vue organisation »

4.1.3 .Vue informationnelle

La troisième vue, cherche à préciser les données manipulées par les rôles pour accomplir les activités d'un processus.

Le paquetage suivant illustre les composants de cette vue :

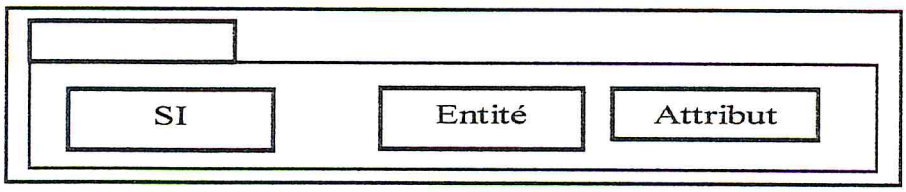


Figure III-22: La vue des composants « vue données manipulées »

Le diagramme de classe suivant présente les différents composant de cette vue d'une manière détaillée :

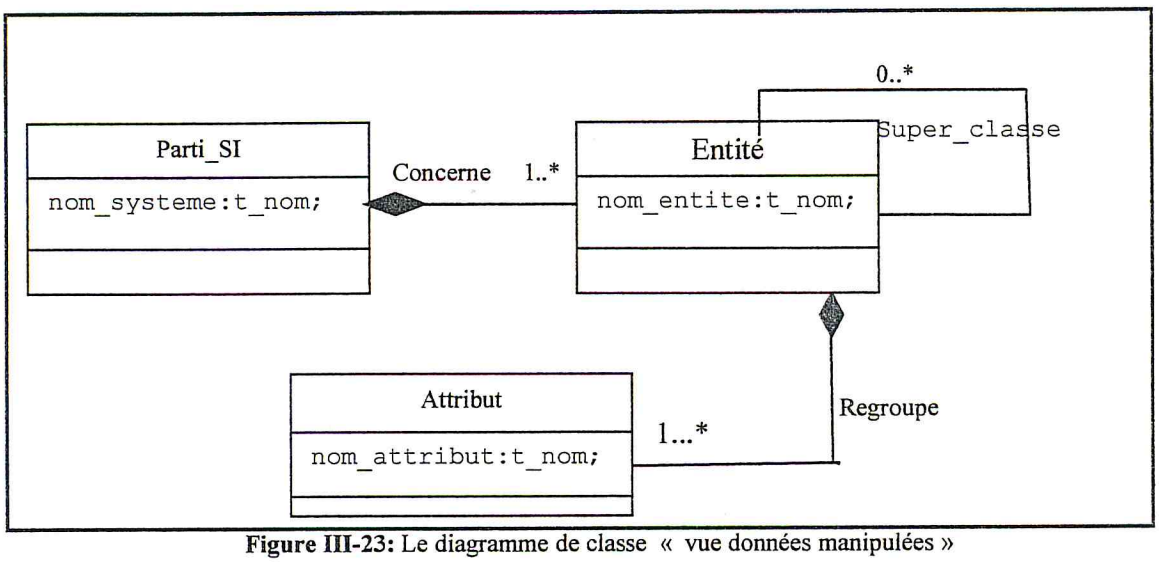


Figure III-23: Le diagramme de classe « vue données manipulées »

Le diagramme suivant illustre la relation entre les différentes vues de processus :

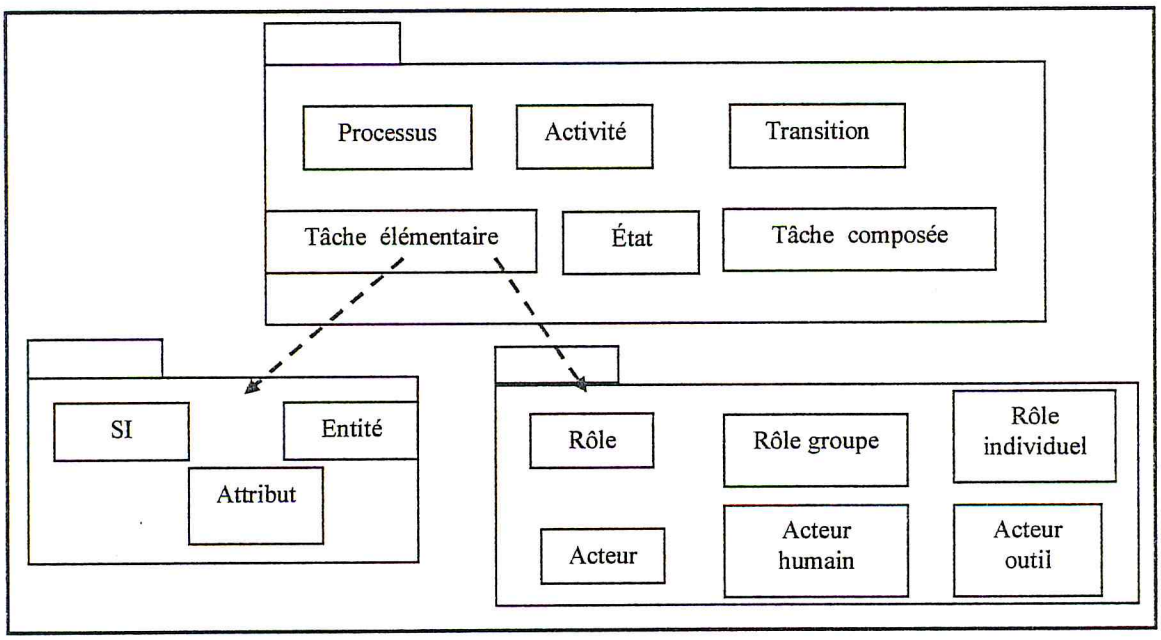


Figure III-24: Vue globale du méta modèle

Le diagramme de classe suivant contient toutes les entités de toutes les vues, et les relations entre elles. Ce diagramme représente le méta modèle.

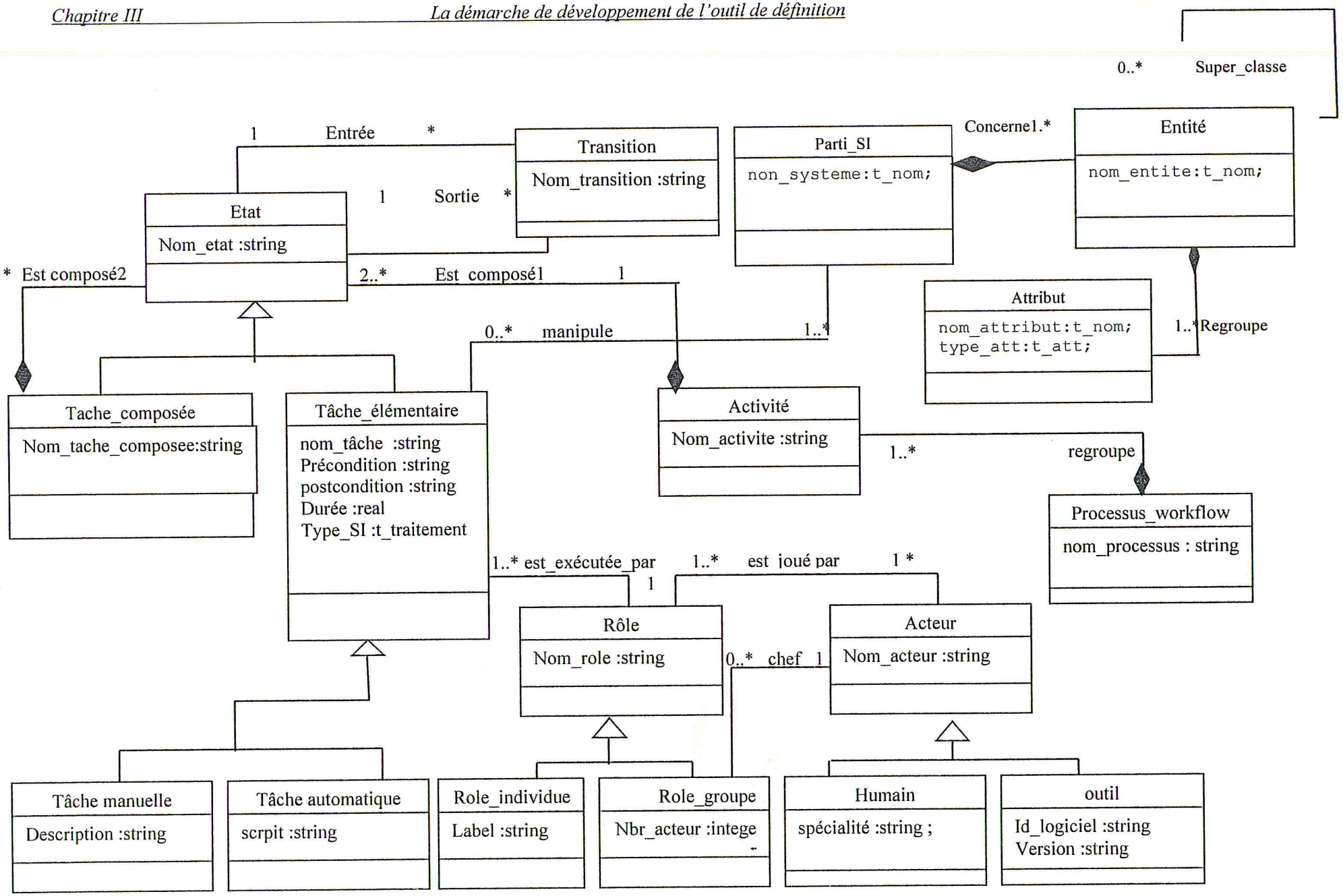


Figure III-25: Vue globale d'un processus du workflow

Nous constatons après l'analyse du méta modèle précédent qu'un *processus* du workflow est composé de plusieurs *activités*. Une *activité* est un ensemble *d'états* muni d'une relation d'ordre exprimée par la classe *transition*. *L'état* se spécialise en deux sous classes : *tâche élémentaire* et *tâche composée* qui est de sa part composée de plusieurs *états*.

La *tâche élémentaire* est accomplie par un *rôle* qui peut être joué par plusieurs acteurs dont un est élu comme chef. Un *acteur* peut être soit, un être *humain* soit un *outil*, cette diversité est exprimée par la spécialisation de l'acteur en humain et outil.

4.2 . Diagramme d'états-transitions

Ce diagramme a pour but de présenter l'enchaînement des différents états que peut prendre une classe, à la suite de traitement particulier. Un diagramme d'états-transitions concerne donc une classe particulière en représentant tous les états possibles que peut prendre cette classe et les évènements (traitements) qui provoque un changement d'état de la classe [Muller, 01]. Nous utiliserons ce type de diagramme pour représenter les différents états que peuvent prendre un processus (Figure III-26) et une tâche élémentaire (Figure III-27).

Commençons par le diagramme d'états transitions pour la classe processus. Les états décrits ci-dessous doivent permettre de gérer le processus. Nous dénombrons 5 états distincts :

- **Initialisé:** cet état correspond à l'identification d'un nouveau processus, soit du fait de la définition d'une organisation, soit du fait de la définition des données.
- **En création:** cet état décrit la phase de construction d'un processus.
- **Enregistré:** quand un processus est enregistré, une définition lui est généré.
- **Défini:** un processus est défini lorsqu'il existe sous une forme graphique et textuelle.
- **Modifié:** cet état permet de gérer une nouvelle version d'un processus suite à une demande de modification.

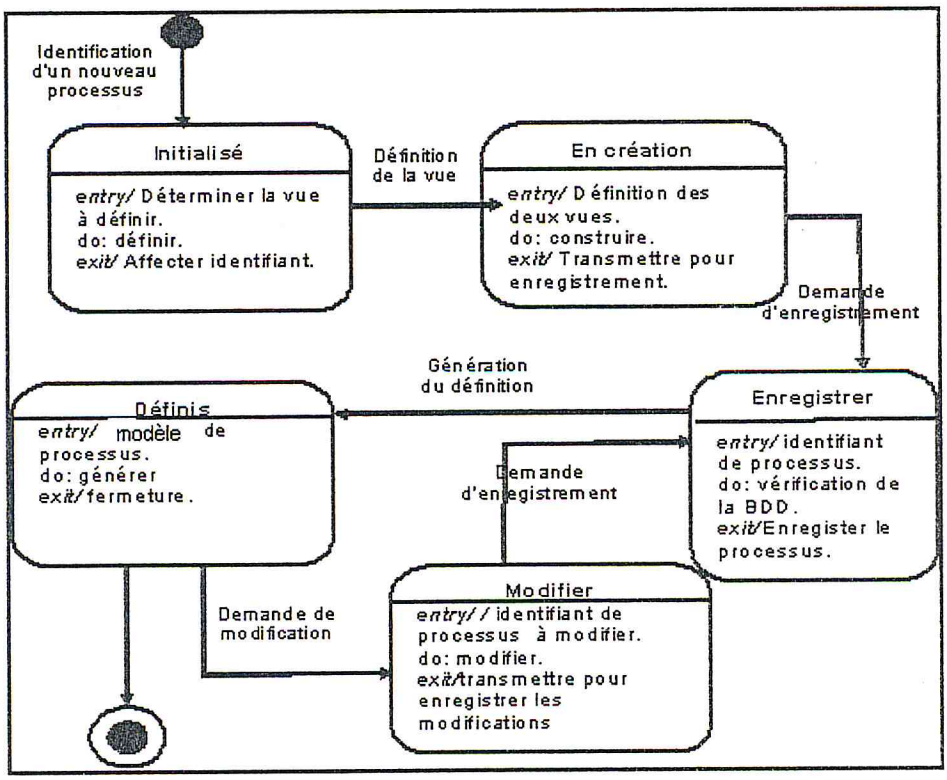


Figure III-26: Le diagramme d'activité pour la classe processus

Le deuxième diagramme d'état transitions pour la classe tâche élémentaire. Une tâche est décrite par les états présentés ci-dessous :

- **En création:** cet état décrit la phase de création d'une tâche.
- **Défini:** une tâche est définie lorsque ses propriétés sont remplies.
- **Enregistré:** une tâche est enregistrée quand elle est définie.
- **Modifié:** cet état permet de modifier les propriétés d'une tâche suite à une demande de modification.
- **Supprimé :** cet état permet de supprimer une tâche déjà crée.

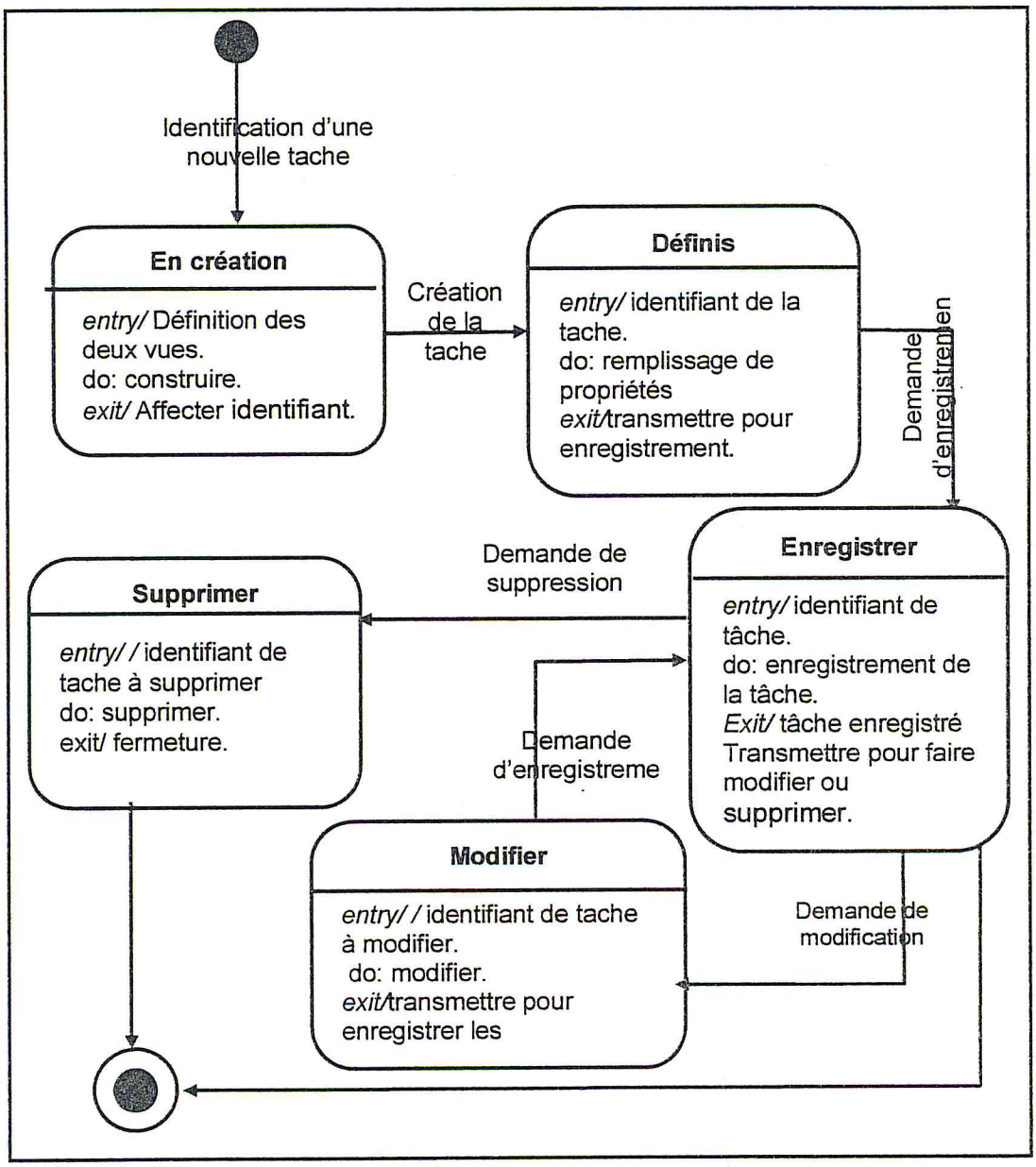


Figure III-27: Le diagramme d'activité pour la classe tâche

5. La conception

La conception s'intéresse d'abord au « comment », à savoir la solution du problème énoncé. Elle commence par une conception dite « globale » qui décrit l'architecture du système, elle se poursuit par une conception détaillée. Pour réaliser la conception, UML propose les diagrammes de classes, de composants et les paquetages.

5.1 . Conception globale

La conception globale a pour but de décomposer le logiciel en modules et de préciser les interfaces et les fonctions de chaque module. A l'issue de cette étape, on obtient une description de l'architecture du logiciel et un ensemble de spécifications de ses divers composants [Bres,93].

L'architecture de l'outil que nous proposons a été élaborée en vue de la communication entre notre outil ODWF, et un contrôleur de modèles EXPRESS nommé Ecco. La figure suivante donne une vision globale de l'architecture de l'outil.

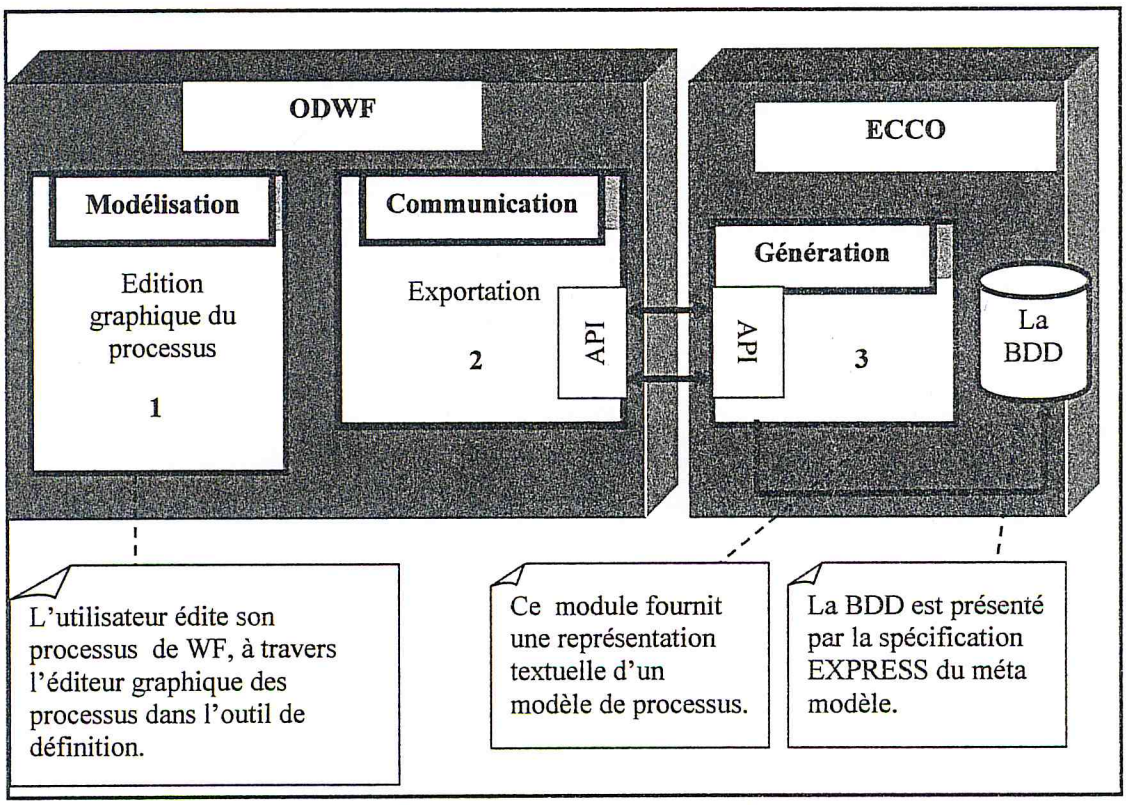


Figure III-28: L'architecture de ODWF

Dans cette architecture, nous avons défini une interface de communication permettant d'envoyer l'information et de récupérer des données du contrôleur de modèles Ecco. Ce dernier fournit une interface d'accès à la base de données.

Afin de mettre en œuvre notre outil, nous avons défini une architecture logicielle composée de trois (3) modules:

- Module 1 : la modélisation des processus.
- Module 2 : la communication.
- Module 3 : la génération des instances.

Le module de génération des instances correspond à un générateur des représentations textuelles. La modélisation est réalisée à travers le module de modélisation des processus, et le

module de communication qui assure la communication entre les deux modules, à travers l'exportation des données depuis l'outil vers la BDD[14]. Le module de communication garantit la correspondance entre la représentation graphique et la représentation textuelle du processus défini par notre outil.

Lors de la modélisation, les données de chaque objet modélisé seront envoyés via le module de communication au générateur, ce dernier enregistre les informations dans la BDD, et lors de la génération du définition, toutes les données de la BDD sont sauvegardées dans un fichier textuel, ce dernier correspond aux fichier physique généré par Ecco.

Le premier module « modélisation » nous donne comme résultat un fichier possédant une extention (.dia) indiquant la représentation graphique d'un processus modélisé. Le deuxième module « génération » produit pour chaque processus modélisé, un fichier textuel sous une extention (.spf). Le dernier module « communication » a pour but d'exporter les données, il utilise une bibliothèque dynamique (.dll) produite par Ecco qui permet la manipulation des données.

5.2 . Conception détaillée

La conception détaillée fournit pour chaque module une description détaillée de la manière dont les fonctions du composant sont réalisées: algorithmes, représentation des données [Sommerville, 88].

5.2.1 .Module 1 : modélisation des processus

Les méthodes de modélisation permettent de construire des modèles à partir d'éléments de modélisation qui constituent des concepts fondamentaux pour la représentation du système [Muller, 97].

Les éléments de modélisation que nous avons utilisé pour construire le modèle de processus, sont représentés dans le paquetage suivant :

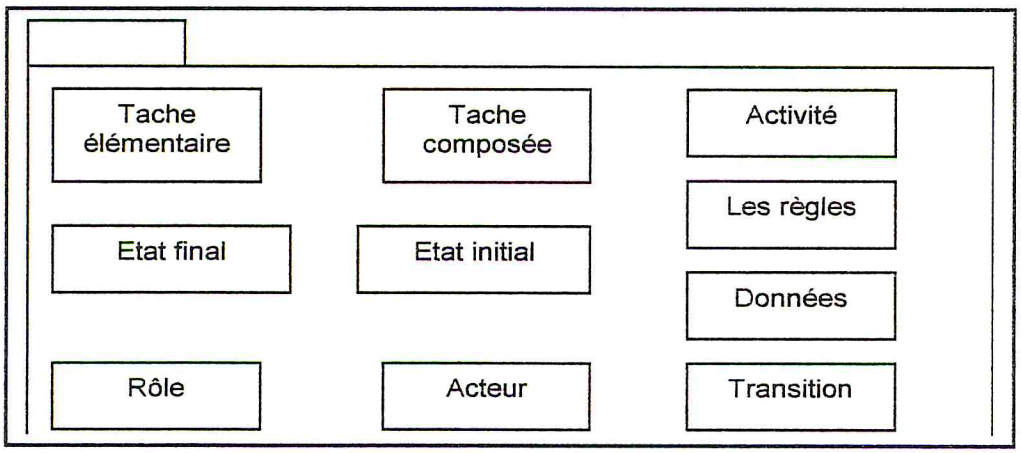


Figure III-29: Les éléments de modélisation

La principale fonction de notre outil est la modélisation, modéliser un processus consiste à le représenter sous une forme d'automate intégrant des tâches, des conditions de déclenchement et d'arrêt et des informations associées à chaque activité (tâches, rôles, données manipulées, ...etc.). Pour réaliser cette fonction, il faut définir une notation graphique [Hamdah, 03] avec laquelle on définira le modèle de processus. Dans ce qui suit nous présenterons les symboles de notation que nous utilisons :

[14] **BDD** : Base De Données

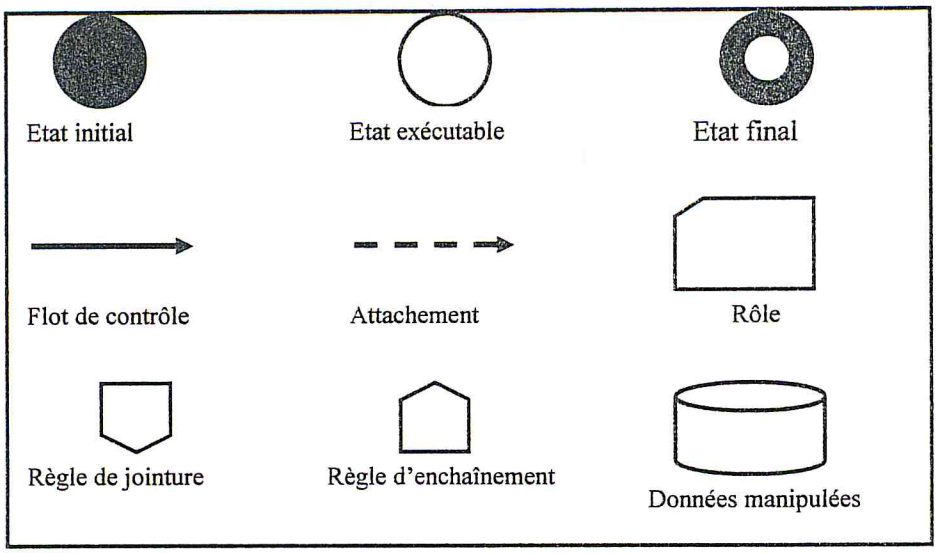


Figure III-30: Les symboles de modèle de processus

Etat initial: chaque modèle de processus de workflow doit contenir exactement un objet *Début* du processus qui représente la tâche initiale. Un objet de la classe *Début* du processus n'a aucun prédécesseur et a un ou des successeurs.

Etat final : la *Fin* représente la tâche finale d'un processus. Un objet de la classe *Fin* marque la fin d'un processus de workflow. Cet état n'existe qu'une fois dans un modèle de processus. L'objet *Fin* a au moins un prédécesseur et n'a pas de successeur.

La tâche : une tâche est représentée par un cercle étiquetée par le nom de l'état. Ce symbole permet de représenter les deux types de tâches : élémentaire et complexe.

Le flux de contrôle : représente le flux de contrôle entre les tâches. Il permet d'exprimer la relation précédence entre deux tâches distinctes. Il est représenté par un arc reliant un état à son successeur suivant une règle d'enchaînement.

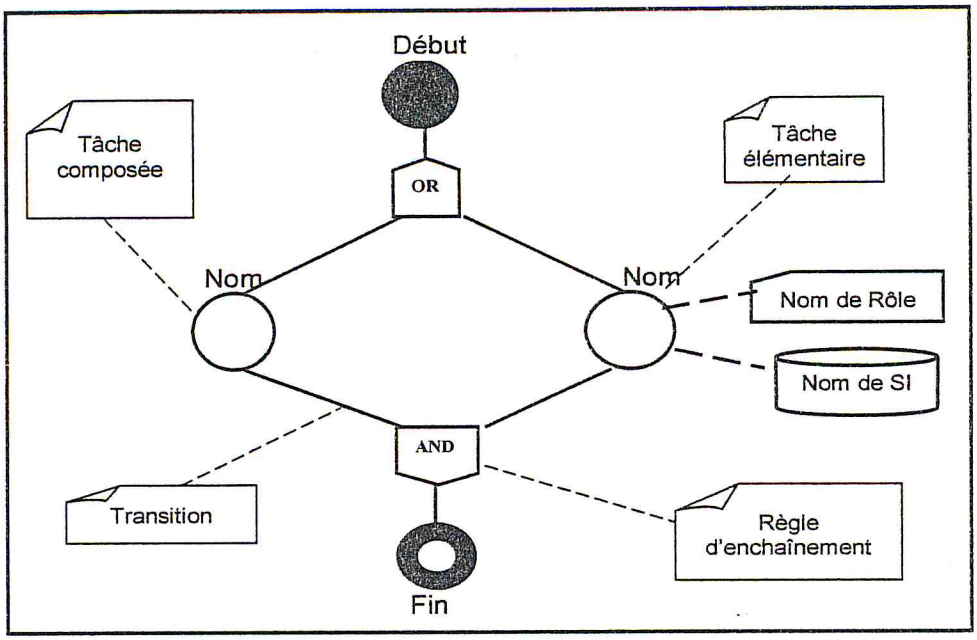
Affectation de rôles : représente l'affectation d'une tâche à son rôle correspondant. Elle est représentée par un arc avec une ligne discontinue, allant de la tâche élémentaire vers le rôle approprié. Notons que les tâches complexes ne possèdent pas de rôles.

Le rôle : un rôle est représenté par un polygone, étiqueté par le nom du rôle. Pour la clarté du modèle, on peut dupliquer la représentation d'un rôle dans un même modèle de processus.

La partie de SI : elle désigne une partie du SI invoqué par l'exécution d'une tâche exécutable. Elle est représentée par un cylindre étiqueté par le nom du sous système. Comme le cas d'un rôle, une tâche complexe n'a pas de système approprié.

La règle d'enchaînement : la règle d'enchaînement est une expression booléenne, décrivant le type de la relation qui existe entre deux tâches adjacentes. Elle permet de représenter deux types de relation OR, AND qui décrivent respectivement, la relation de l'alternance et celle du parallélisme.

La règle de jointure (synchronisation) : la règle de jointure permet, de faire la jointure de plusieurs tâches pour enchaîner avec une nouvelle tâche. Elle permet de représenter deux types de relation OR, AND. La figure suivante montre un exemple d'instanciation du méta modèle présenté précédemment :



FigureIII-31: Un exemple de la notation utilisée.

A chaque tâche élémentaire on associé un rôle et une partie de SI, les tâches sont reliées les unes aux autres par des liens qui représentent des transitions. Chacun élément modélisé est accompagné d'une fiche descriptive chargée d'en compléter ses données.

Les diagrammes de classe et de composants suivants représentent les éléments de modélisation qui doivent être implémentés, d'une manière synthétique :

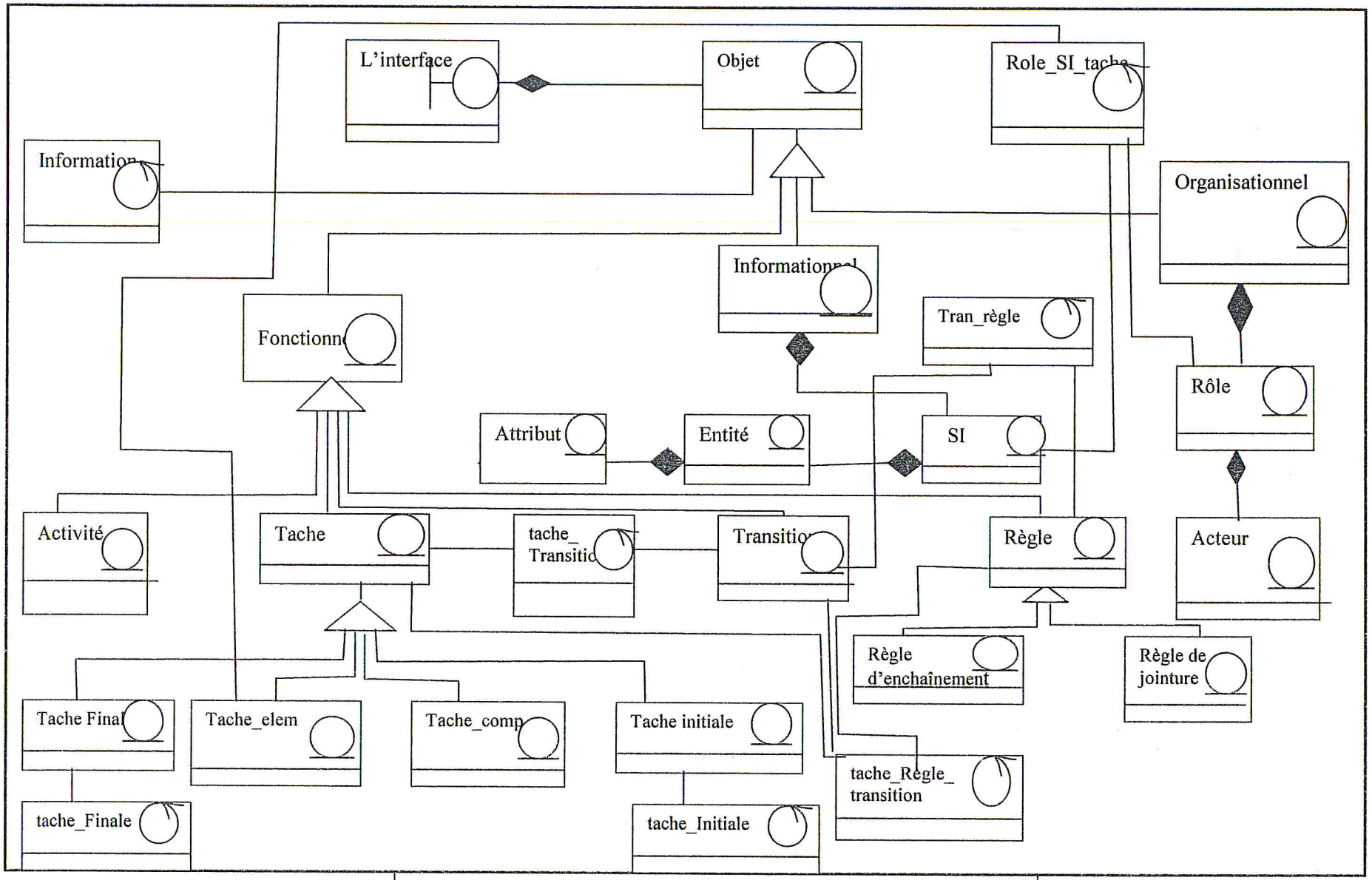


Figure III-32 : Le diagramme de classe des éléments de modélisation

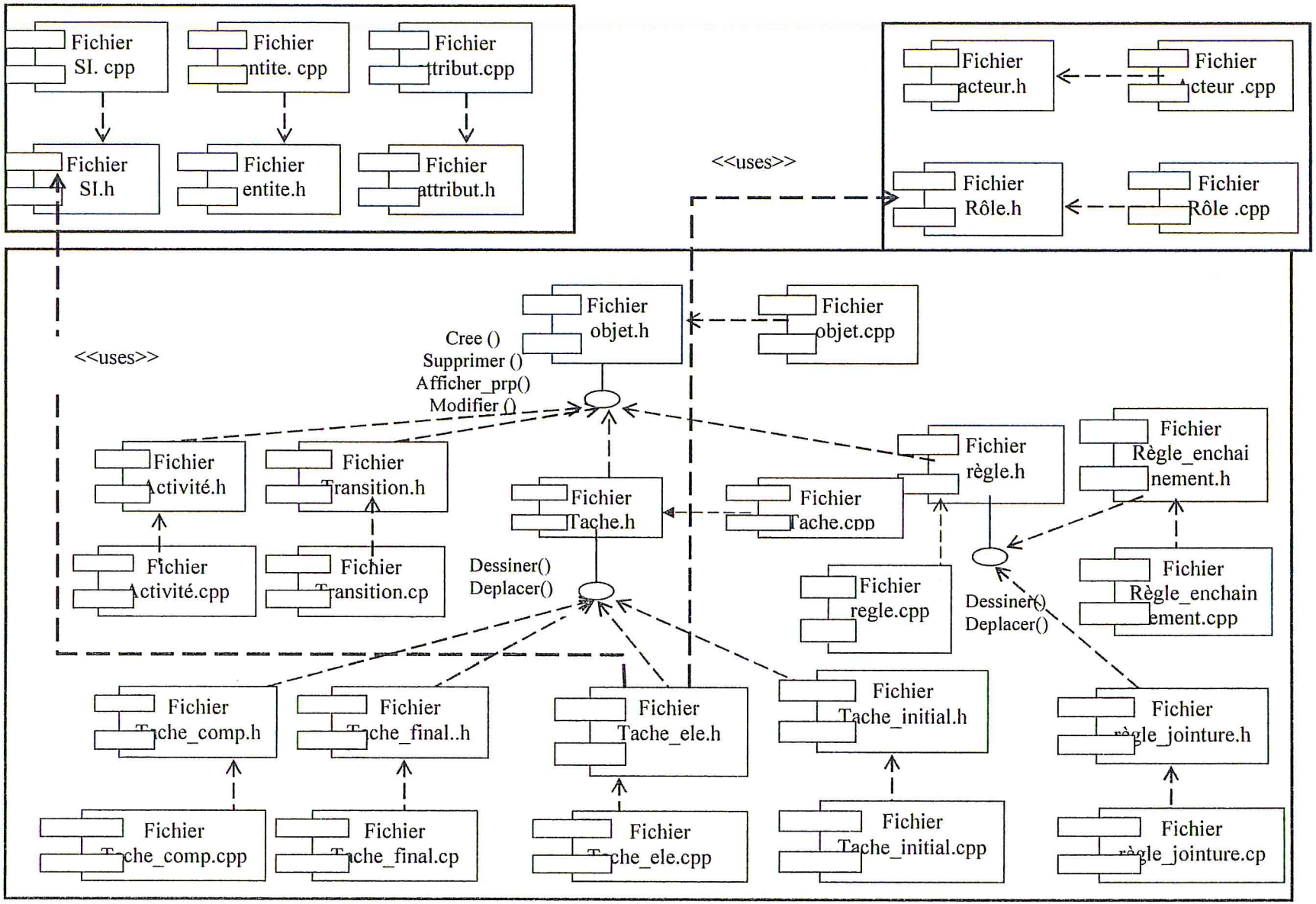


Figure III-33: Le diagramme de composants pour les entités implémentées

Dans le diagramme de composant chaque classe est réalisée par deux composants: la spécification et le corps, la spécification contient l'interface de la classe alors que le corps contient la réalisation de cette même classe. En C++, la spécification correspond à un fichier avec un suffixe (.h) et le corps à un fichier avec un suffixe (.cpp).

5.2.2 .Module 3 : génération des instances

Ce module est responsable de la génération de la représentation textuelle du processus. Il permet l'interprétation et l'échange des définitions de processus issu de ODWF dans le modèle de référence par un moteur de workflow (présenté dans le chapitre I). Les documents qui vont être échangés (dans un format textuel) dans l'interface d'import/export véhiculeront les types d'information tels que :

- Conditions de déclenchement et de terminaison de processus.
- Identification d'activités dans le processus incluant les applications externes associées.
- Identification des types de données et des chemins d'accès.
- Définition des conditions de transition et des règles de routage.

Le problème qui se pose comment générer une représentation textuelle associée à une représentation graphique d'un processus?

Pour résoudre ce problème, il faut que le fichier généré par notre outil soit conforme au méta modèle présenté dans la section précédente. Pour cela nous avons interprété ce méta modèle avec le langage EXPRESS, et l'instanciation de schéma EXPRESS de méta modèle correspond à la représentation textuelle du modèle.

La solution que nous proposons au problème de génération comporte deux étapes :

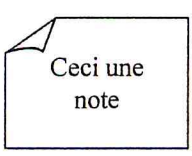
1. **la spécification**: dans cette étape, il faut faire un passage d'UML vers EXPRESS, pour développer le schéma EXPRESS du méta modèle.

2. **l'instanciation**: cette étape nécessite un outil qui permet l'instanciation de schéma EXPRESS du méta modèle issue de l'étape de spécification. Ceci est possible (comme mentionné dans le chapitre précédent) puisque le langage EXPRESS est traitable par machine.

5.2.2.1 . La spécification du méta modèle avec le langage EXPRESS

Le méta modèle précédant est décrit en UML, ce dernier est un langage semi-formel, qui n'est pas traitable par la machine, cependant, nous avons utilisé le langage EXPRESS pour spécifier le méta modèle.

Pour passer depuis les concepts du langage UML que nous avons utilisé dans le méta modèle vers ceux du langage EXPRESS,nous étions obligées de définir des règles de passage, cela a été fait après étude des deux langages, ces règles sont résumées dans le tableau suivant :

	UML	EXPRESS
Les notes		Les commentaires délimités par (*...*). Un commentaire sur une ligne est tout ce qui suit "--" jusqu'à la fin de ligne.

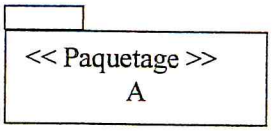
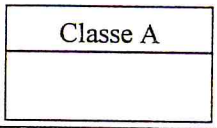
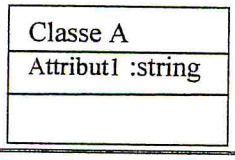
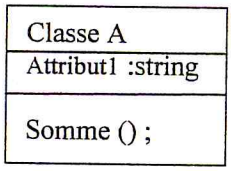
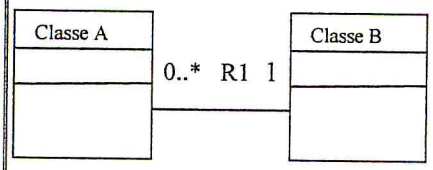
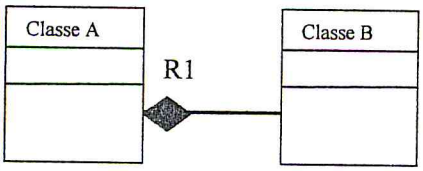
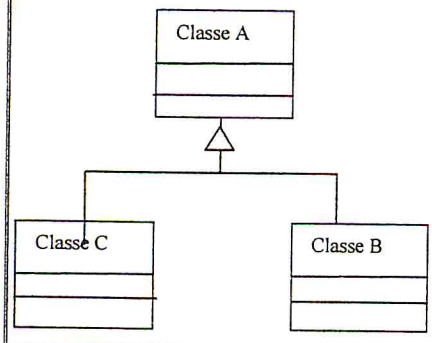
Les paquetages		**SCHEMA** A ; .... **END_SCHEMA** ;
Les classes		**ENTITY** A .... **END_ENTITY**;
Les attributs		**ENTITY** A Attribut1 :STRING ; ..... **END_ENTITY**;
Les fonctions		**ENTITY** A Attribut1 :string ; ..... **DERIVE** somme :.... :=..... ; **END_ENTITY**;
Les cardinalités		**ENTITY** A R1 : B; ..... **END_ENTITY**; **ENTITY** B ..... INVERSE R : SET[0:?] OF A FOR R1; **END_ENTITY**
L'agrégation		**ENTITY** A R :LIST OF B; ..... **END_ENTITY**; **ENTITY** B ..... INVERSE R1 : A FOR R; **END_ENTITY**
L'héritage		**ENTITY** A ABSTRACT SUPERTYPE OF (ONEOF /ONEOR /AND (B, C)); ..... **END_ENTITY**; **ENTITY** B SUBTYPE OF (A); ..... **END_ENTITY**;

Tableau III-1: Les règles de passage des concepts UML vers EXPRESS

La spécification EXPRESS du méta modèle représente la base de données, qui sera créé à partir des règles présentées ci dessus.

5.2.2.2 . Linstanciation des schémas EXPRESS

Avec les formalismes de spécification, et en particulier avec le langage EXPRESS, sont apparus des outils permettant de vérifier qu'une certaine population de données respecte une spécification. Ces outils sont appelés "contrôleurs de modèles ".

Les contrôleurs de modèles sont des outils chargés d'offrir des services de création et d'édition de spécifications de données, et aussi de création, de contrôle et de test de populations de données conformes à une spécification [Sardet, 99].

Parmi ces contrôleurs, notre choix est porté sur l'utilisation de Ecco Toolkit, ce dernier est abordé d'une manière détaillée dans l'annexe C.

L'instanciation d'un modèle de données et le contrôle de cette instanciation peuvent se faire, sous Ecco, de deux façons distinctes :

→ Soit en utilisant une interface de contrôle générique (fournie par l'outil) permettant de créer des instances, de les mettre à jour et de vérifier les règles d'intégrité définies dans le modèle ;

→ Soit en créant une application en charge de réaliser ces mêmes tâches, mais dans un contexte applicatif donné.

La Figure suivante illustre la mise en œuvre de ces deux approches avec l'outil Ecco.

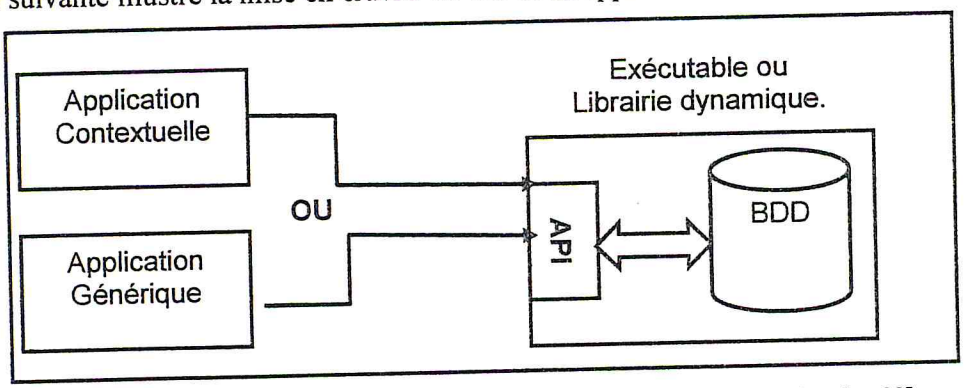


Figure III-34 : Contrôle d'un modèle de données EXPRESS avec Ecco [Sardet, 99]

Dans notre cas nous avons choisi à la deuxième solution, c'est l'instanciation à partir d'une application contextuelle, cette dernière est représentée par le module de modélisation.

Notre choix été guidé par :

- L'assurance de la correspondance entre le modèle défini dans le module de modélisation, et la représentation textuelle générée par le module de génération des populations d'instance.
- La première façon d'instanciation amenait l'utilisateur à maîtriser l'Ecco et comprendre son fonctionnement, ce qui rendre notre outil inefficace.

5.2.3 .Module 2 : la communication

L'API[15] du module de communication doit permettre de réaliser sur la base de données Ecco toutes les interrogations et mises à jour possibles, cela en fonction des limitations de l'API Ecco

[15] **API** : Application Programming Interface

elle même. Nous soulignons deux types de connexions de l'Ecco, qui sont attachées au adaptateur.

5.2.3.1 . Connexion avec adaptateur

L'API ECCO est l'API de communication avec la base de données fournit un ensemble de primitives permettant de manipuler la base [Ecco, 01], c'est-à-dire de réaliser des mises à jour, de créer ou de détruire des instances, de valuer ou de modifier des attributs, ...

Cette API Ecco est accessible au moyen d'un adaptateur de langages. Cet adaptateur (Figure III-35) consiste à permettre à toute application développée dans un langage donné (C, C++, Java, Tcl/Tk, Visual Basic) [Ecco, 01], d'accéder au code exécutable (ou la bibliothèque dynamique) généré par Ecco, pour manipuler la BDD.

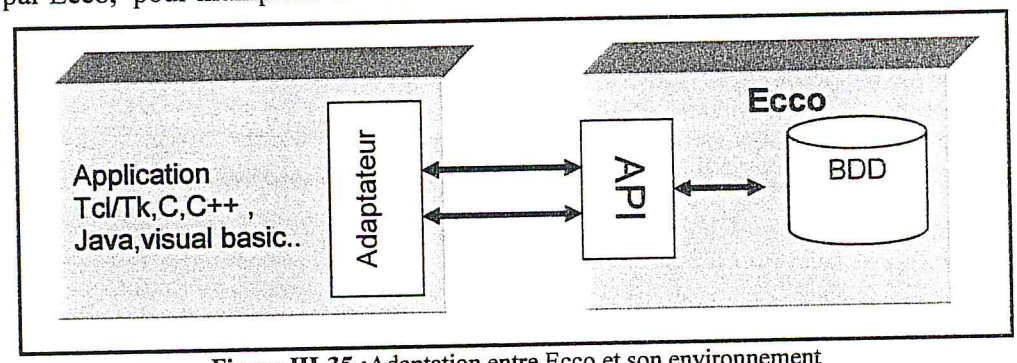


Figure III-35 : Adaptation entre Ecco et son environnement

Pour l'outil développé, nous avons utilisé, pour des raisons de commodité, le langage C++ pour la communication de notre application avec le module « génération du population d'instances». La création d'une instance dans la base de données Ecco se fera de la façon suivante :

Application C++	Instance* in=new Instance(TypeInfo(**app**, " NOM_*SCHEMA* "," NOM_*ENTITY* ")); in->putAttr("nom", " Tâche1"); in->create();
API Ecco	oid = create('NOM_*SCHEMA*, NOM_*ENTITY* , 'Tâche1')
Base de données	#1 = NOM_*ENTITY* (" Tâche1 " ,$,$)

Adaptateur

SDAI

Tableau III-2: Création d'une instance dans Ecco

Une commande de création issue de l'adaptateur de langages est envoyée à Ecco. Cette commande est une ensemble d'instructions représentant l'action à réaliser conformément à la spécification de l'API Ecco (création d'une instance de type NOM_*ENTITY*, dans le modèle de données portant le nom NOM_*SCHEMA*). Finalement, une instance, est créée dans la base de données, et son identificateur (*#1*) retourné à l'application appelante.

Le diagramme de composants suivant représente les différents composants qui intervenant à la communication entre les deux modules : la génération et la modélisation, en utilisant la connexion avec adaptateur.

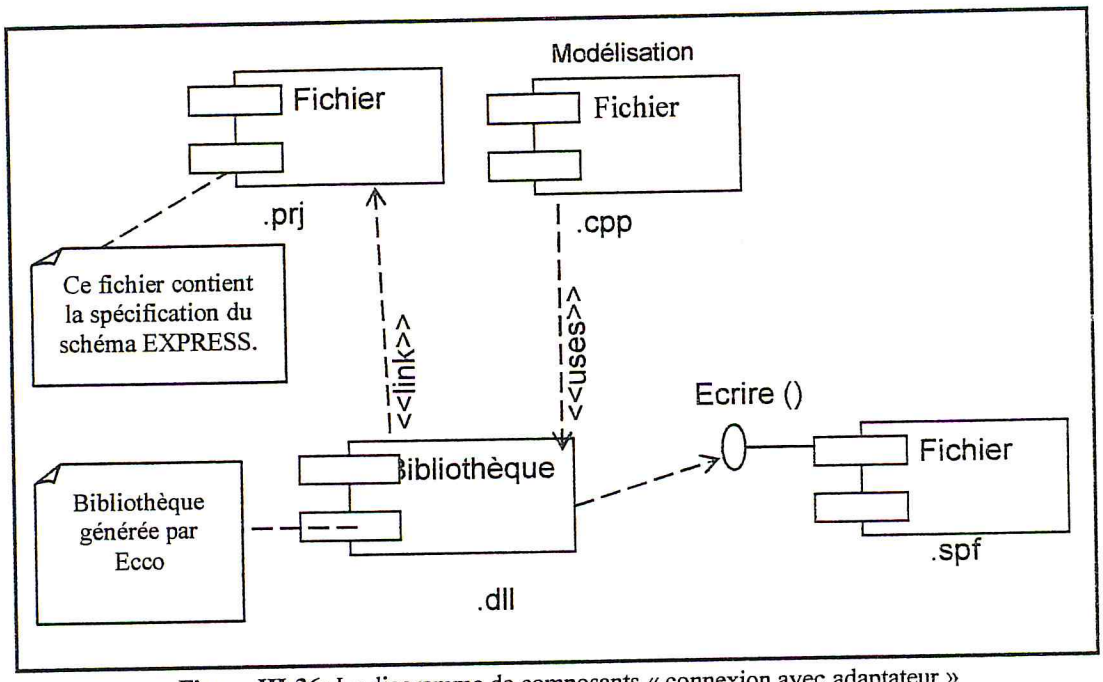


Figure III-36: Le diagramme de composants « connexion avec adaptateur »

Dans ce diagramme le schéma EXPRESS est présenté par le fichier (.prj), une fois la compilation de ce fichier par Ecco est terminé, une bibliothèque (.dll) sera générée, cette bibliothèque est utilisée par le composant de modélisation pour l'écriture de fichier physique, qui est présenté par le fichier (.spf).

5.2.3.2 . Connexion sans adaptateur

Nous soulignons dans ce cas que ODWF et le contrôleur Ecco possédaient chacun une API de communication. Afin que la communication puisse s'établir, il reste donc à les interconnecter.

L'approche que nous proposons dans ce cas est basée sur l'utilisation d'une application passerelle. En effet, nous avons vu que le contrôleur de modèles était utilisable par l'intermédiaire d'un adaptateur de langages, permettant ainsi des développements sous différents types d'environnements (Tcl/Tk, C++, C, Java...). Or, il n'existe aucun adaptateur de langages pour le langage de programmation du module de modélisation.

Cependant, le langage de programmation du module de modélisation permet d'appeler des primitives externes, développées dans d'autres environnements (Tcl/Tk, C++, C...), à condition que celles-ci soient disponibles sous la forme de librairies dynamiques (DLL sous Windows).

Nous sommes donc obligées d'utiliser dans ce cas, le concept de librairie dynamique, afin de rendre possible des échanges entre l'éditeur et le contrôleur.

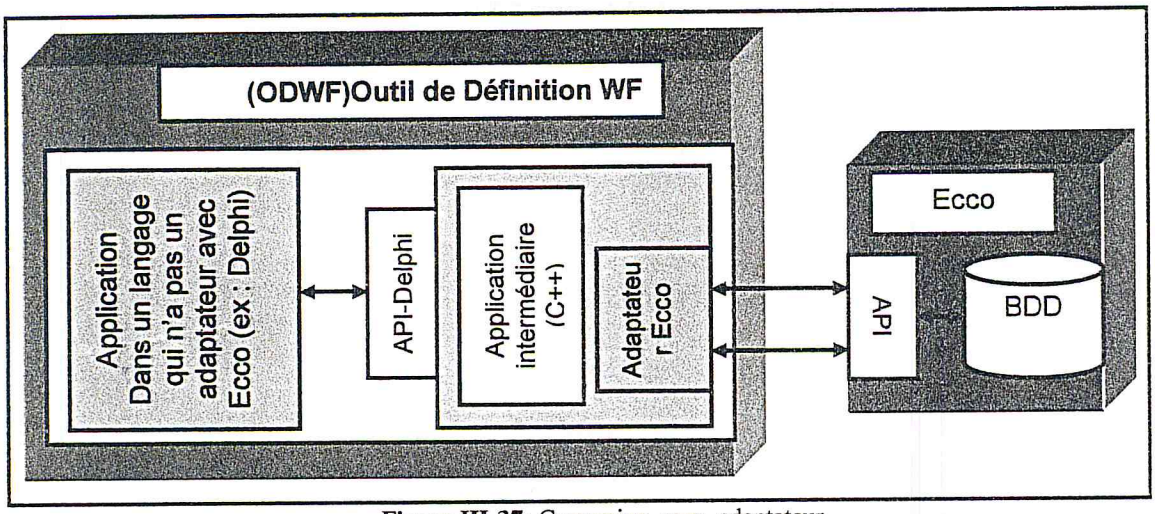


Figure III-37: Connexion sans adaptateur

Dans le cas présenté ci-dessus, le compilateur Delphi est donc en charge de la gestion de l'adaptateur d'ODWF vers Ecco. Cet adaptateur est construit sur un compilateur (C++), c'est donc au travers de ce dernier que les commandes issues de Delphi seront envoyées, via l'API-Delphi, à l'application passerelle de C++, celui-ci transmettra l'information, au travers de l'adaptateur Ecco et via l'API Ecco à la base de données.

Le diagramme de composants suivant représente les différents composants pour établir la communication entre le module de modélisation et celui de génération, en utilisant la connexion sans adaptateur.

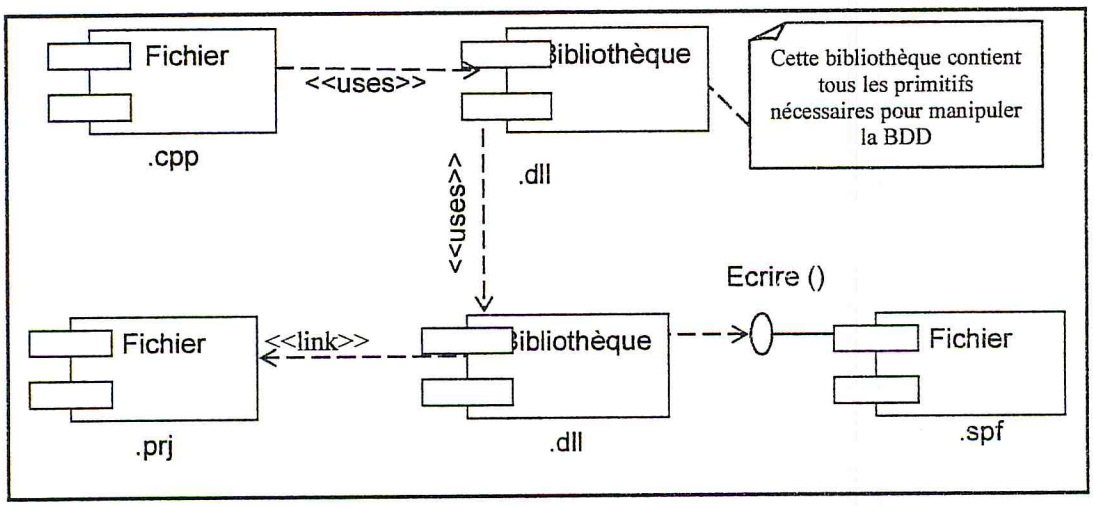


Figure III-38: Diagramme de composants « connexion sans adaptateur »

Dans ce cas le composant de modélisation utilise une bibliothèque développée dans un environnement (Tcl/Tk,C++,C…), cette bibliothèque fait appel à une autre bibliothèque produite par l'exécution du schéma EXPRESS par Ecco, pour la génération de fichier physique qui est présenté par le fichier (.spf).

5.2.3.3 . Le choix de la méthode de connexion

Notre choix c'est donc porté sur la première possibilité à cause de la souplesse et de la simplicité de la connexion qui n'a pas besoin d'une application passerelle.

6. L'implémentation

A ce niveau, il s'agit d'implémenter la solution retenue au niveau de la phase de conception, c'est la phase au cours de laquelle les structures et les algorithmes définis pendant la conception sont traduits dans un langage de programmation et / ou une BDD.

6.1 . Choix du langage de programmation

Une des décisions les plus importantes dans la conception et la réalisation d'un logiciel est le choix du langage de programmation. Après le choix du type de connexion, nous pouvons maintenant choisir le langage de programmation. Dans notre cas le langage de programmation devait être choisi parmi les langages suivants : C, C++, java et visual Basic. Notre choix s'est porté sur le langage C++ parce que c'est un langage orienté objet, et la programmation dans un langage objet est la manière la plus commode de traduire une conception objet en un langage exécutable par machine [Muller, 97].

6.2 . Création de la base de données

L'objectif primordial de la modélisation conceptuelle de données est d'aboutir à un modèle logique de données qui permettra ensuite la création d'une base de données [Dehaninsala, 02].

Chaque approche de modélisation est plus adaptée à une catégorie spécifique de base de données. C'est ainsi que les modèles entité-association sont plus destinés aux bases de données relationnelles et les modèles conceptuels objets sont les mieux adaptés aux bases de données objets étant donné qu'ils ont des concepts similaires (héritages, polymorphisme, ...etc.). Dans notre projet nous avons développé une BDD objet, utilisant EXPRESS, et son environnement Ecco.

Pour créer notre base de données nous avons respecté les règles de passage de UML vers EXPRESS décrites précédemment. Parmi les différentes entités du méta modèle nous avons spécifié quelques entités par le langage EXPRESS dans la section suivante, et l'annexe A comporte la spécification complète du méta modèle.

➢ **Processus :** est décrit par l'entité suivante :

```
ENTITY processus;
        Nom_processus : t_nom;
        Activites :LIST OF activite;
    UNIQUE
        Ur :nom_processus;
    END_ENTITY;--processus

    TYPE t_nom = STRING;
        WHERE LENGTH(SELF)>0;
END_TYPE;
END_ENTITY
```

Figure III-39. Définition d'un processus

Un processus est défini par un nom, *Nom_processus,* de type, *t_nom* et une liste d'activités, *Activites* de type, *activite*. Notons que le type, *t_nom*est défini à partir du type de base, *STRING.*

➢ **Activité :** comme nous l'avons vu, une activité est une liste d'états. Le code qui permet de décrire une activité est le suivant :

```
ENTITY activite;
        nom_activite : t_nom;-- c'est le type déclaré précédemment
        etats : LIST OF etat;
INVERSE
        compose   : processus FOR activites;
UNIQUE
        Ur :nom_activite;
END  ENTITY ;--activite
```

Figure III-40: Définition d'une activité

Une activité est vue comme étant un automate formé d'une liste d'états, *etats* représentant les tâches, reliés par des liens de précédence.

Une activité est caractérisée par un nom unique, *nom_activite* de type *t_nom*. L'attribut inverse, *compose* permet de relier l'activité au processus dont elle fait partie.

➢ **Etat :** Un état représente le contexte pour le quel passe une activité. Formellement, un état est décrit par l'entité suivante:

```
ENTITY etat
ABSTRACT SUPERTYPE  OF (ONEOF(tache _elementaire, tache _composee));
        nom_etat : t_nom;

INVERSE
        transition_entree  :   SET [0 :?] OF transition FOR etat_sortie;
        transition_sortie  :   SET [0 :?] OF transition FOR etat_entree;
        compose_1          :   activite FOR etats;
        compose_2          :   etat_composee FOR etatss;
UNIQUE
        Ur :nom_etat;
END_ENTITY; --etat
```

Figure III-41: Description d'un état

Un état est une généralisation de deux sous classes, *tâche_élémentaire* ou exécutable et *tâche_composee* dont elle nécessite une décomposition.

Un état est une entité caractérisée par un nom, *nom_etat* de type *t_nom* et un ensemble d'attributs inverses, *transition_entree*, *transition_sortie*, *compose_1* et compose_2 qui décrivent respectivement la transition entrante, celle sortante, l'activité dont la quelle appartient ainsi que la *tâche_composee* composée par cet état. La Figure III-42 décrit la tâche élémentaire.

```
ENTITY tache_elementaire
    ABSTRACT SUPERTYPE OF (ONEOF (tache_automatique, tache_manuelle ))
    SUBTYPE OF  (etat);
   Nom_tache_elementare :STRING ;
    Pre condition   : STRING ;
    Post condition  : STRING ;
    Est_accomplie_par : role;
    Duree :REAL;
    Donnees : partie_SI;
    Type_SI : t_traitement ;
  UNIQUE
         ur :nom_tache_elementaire ;
   WHERE
         wr1 : duree > 0 ;
         wr2 :SIEZOFEOF(QUERY(x <* est_accomplie_par))<> 0;
END_ENTITY ; --tache_elementaire
```

Figure III-42: Tâche élémentaire (exécutable)

Comme nous l'avons vu la classe, *tâche _elementaire* est une classe abstraite pour les deux sous-classes *tâche _automatique* et *tâche manuelle*. Au même lieu est une sous-classe de la classe *etat*. Une tâche élémentaire est caractérisée par une précondition, *precondition* de type chaîne de caractères (string), une description, *description* de type, string, le rôle qui l'accomplie *est_accomplie_par*, les données manipulées *données* ainsi que la durée *Duree* de type réel.

- **Transition:** une transition est une relation permettant d'établir un ordre d'exécution entre deux états successifs. Afin de pouvoir assurer cette fonctionnalité, une transition est définie par l'entité suivante :

```
ENTITY transition;
            nom_transition : t_nom;
            etat_entree   : etat;
            etat_sortie   : etat;
UNIQUE
            Ur :nom_transition;

END ENTITY;--transition
```

Figure III-43 : La définition de l'entité transition

Une transition est caractérisée par un nom unique, *nom_transition*. Elle est caractérisée, également, par deux états, un ancien, *etat_entree* et un nouveau, *etat_sortie*, de type précédemment défini, *etat*.

- **Rôle** : un rôle est l'entité chargée de l'exécution des tâches. En EXPRESS, un rôle est défini comme suit :

```
ENTITY role;
ABSTRACT SUPERTYPE OF (ONEOF(role_individuel, role_groupe));
        nom_role  : t_nom;
        est_joue_par  : SET[1:?] OF Acteur;
        chef :acteur ;
INVERSE
        Accomplie : SET[1:?] OF tache_elementaire FOR est_accomplie_par;
UNIQUE
        Ur : nom_role;
END_ENTITY; -- role
```

Figure III-44: La définition du rôle

Un rôle est décrit par un nom unique, *nom_role,* un ensemble d'acteurs jouant ce rôle et les tâches dont il est responsable. Un rôle peut être spécialisé en deux sous rôles : rôle individuel, *role_individuel* joué par un seul acteur, et un rôle joué par un groupe, *role_groupe,* pour lequel on associe un chef.

6.3 . Les fonctionnalités de l'outil de définition

La démarche de définition de processus se décompose en:

1. Tout d'abord le processus est modélisé par un expert métier dans la notation décrite précedament. Cette description du processus est réalisée à l'aide de notre outil ODWF.
2. Ensuite, une fois la modélisation est terminée, la définition correspondante est générée grâce à Ecco Toolkit. Cette génération est entièrement automatisée et donc immédiate et totalement transparente pour l'utilisateur.
3. Enfin, le workflow peut être exécuté à l'aide du moteur de workflow, cette étape est réalisée par un autre outil de workflow.

6.3.1 .La modélisation

ODWF comprend un éditeur graphique qui permet à un simple utilisateur de concevoir des modèles des processus, de les manipuler pour les modifier et finalement de générer la définition associée au processus modélisé.

Comme nous montre la Figure III-45, l'interface utilisateur est composée de deux fenêtres, chaque fenêtre peut être redimensionnée et adaptée aux besoins de l'utilisateur, ces fenêtres sont:

- Le panneau de modélisation (1)
- Un explorateur de travail (2)

Et trois barres ayant des fonctionnalités spécifiques :

- La barre standard (3)
- La barre d'outils (4)
- La palette des couleurs (5)

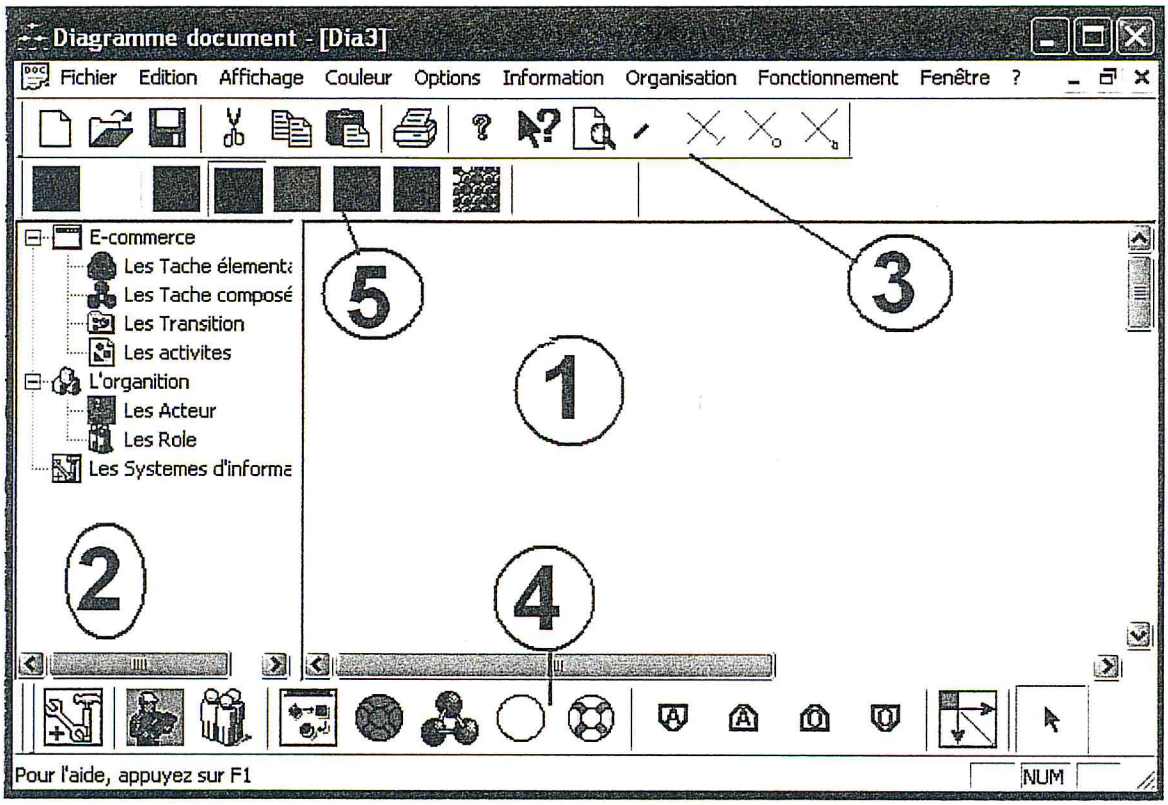


Figure III-45: L'interface de l'outil

6.3.1.1 . Le panneau de modélisation

Cette fenêtre est l'espace du travail dont lequel l'utilisateur va décrire graphiquement son processus à l'aide des éléments graphiques (nœuds, liens, …) de la barre d'outils.

Le graphe de processus est construit en sélectionnant un par un les objets dans la barre d'outils et en les positionnant dans la fenêtre de modélisation. Une boite de dialogue s'affichera après le positionnement de l'objet (ex : tâche) pour remplir les propriétés de cet objet, parmi lesquels on trouve : le nom, le rôle, la durée…etc.

Les fonctions de sélection et déplacement d'un objet standards sont disponibles, et les fonctions de copie et de suppression…etc, sont accessibles par un menu contextuel.

L'ensemble des modèles bénéficie des fonctionnalités suivantes :

→ interface homme/machine aux standards Windows

→ presse-papiers copier/coller

→ zoom, degré de zoom

→ ajustement de taille d'objets

→ Suppression d'un objet

→ sélection et déplacement d'un d'objet

→ impression d'un modèle

6.3.1.2 . L'explorateur

Cette fenêtre est située à gauche de l'écran. L'explorateur présente les objets sous forme d'arborescence (Figure III-46) pour une navigation plus efficace, et pour que nous puissions consulter la liste des tâches, rôles, transitions …etc.

L'explorateur permet la visualisation du contenu de l'environnement de travail. Cet environnement de travail donne à l'utilisateur toutes les fonctions nécessaires pour modéliser son processus. Lors de la modélisation d'un processus, les objets (tâches, transitions, activités, acteurs, rôles..) apparaissent de façon ordonnée dans l'explorateur et sont positionnés au même temps dans le panneau de modélisation.

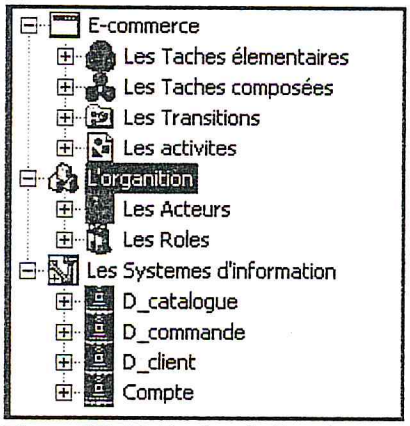
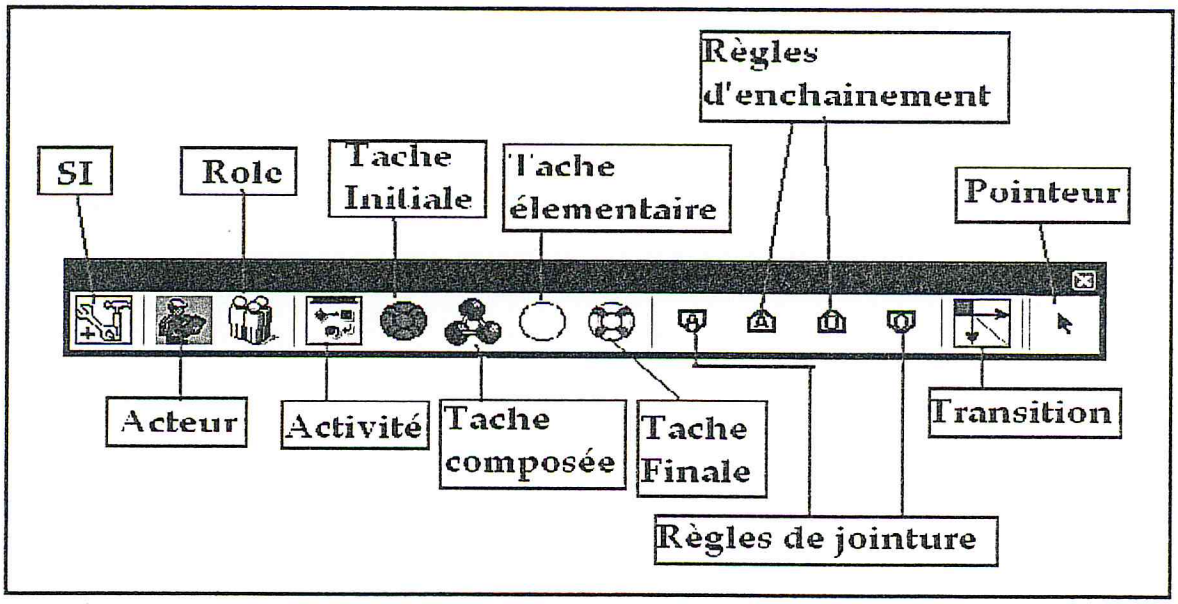


Figure III-46: Fenêtre de l'explorateur

6.3.1.3 . La barre d'outils

La barre d'outils présente les objets utilisés pour la représentation graphique des processus. Elle est déplaçable et peut être positionnée à droite, gauche, haut ou bas de l'espace de travail ou encore en barre flottante dans l'interface.



FigureIII-47: La barre d'outil

6.3.2 .La Génération

ODWF offre les fonctionnalités pour générer automatiquement l'application associée au processus modélisé. Cette génération, ne nécessite aucune compétence informatique, elle est totalement invisible à l'utilisateur. Lorsque l'expert a finalisé la modélisation de son processus, notre outil lui génère un fichier physique. La procédure de génération est extrêmement simple et comprend deux étapes :

- **Le paramétrage** : une boite de dialogue s'affichera, l'ors que l'utilisateur click sur Option de code, dans le menu 'Option' comme nous montre la figure suivante :

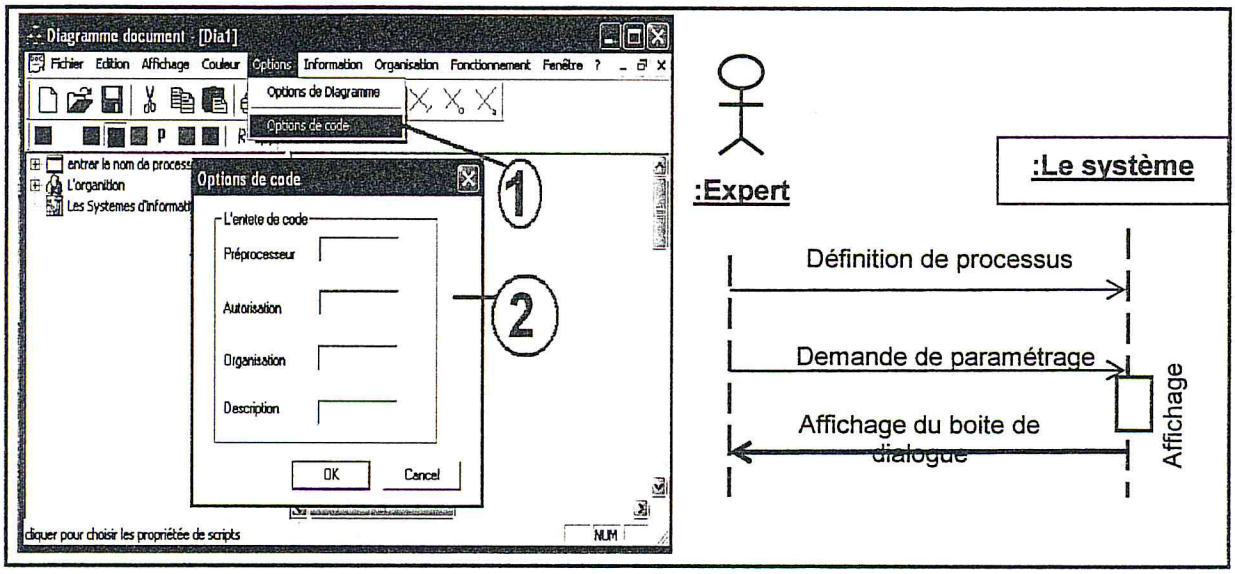


Figure III-48: La fenêtre de paramétrage

Cette boite de dialogue, donne la liste des paramètres à saisir pour l'entête du fichier physique généré parmi lesquels on retrouve, le nom de l'auteur, le nom de l'organisation, l'autorisation,...etc.

- **La génération :** une représentation textuelle sera générée sous le même répertoire et sous le même nom du modèle. Cette représentation est correspond aux fichier physique générer par Ecco.

```
#1=TACHE_AUTOMATIQUE('Consulter', 'Achat electronique', 'commande de
produit', 'consultation du site', '5:11:44', 'Abonn\S\i',
'D_catalogue', '');
#2=TACHE_AUTOMATIQUE('Inscrire', 'Achat electronique', 'num_compte
valide', 'non inscrire', '04:22:11', 'Client', 'D_client', '');
#3=TACHE_AUTOMATIQUE('Livrer', 'Achat electronique', 'llivraison du
produit au client', 'Som_compte>Som_achat AND facture valide',
'15:44:11', 'Technique', 'Compte', '');
```

Figure III-49: Représentation textuelle d'un processus modélisé

Le fichier physique possède une structure bien définie, présentée dans le chapitre II.

7. Test et validation

7.1 . Le test

Le test permet de réaliser des contrôles pour la qualité du système. Il s'agit de relever les éventuels défauts de conception et de programmation (revue de code, tests des composants,...) [Sommerville, 88].

La validation doit être envisagée lors de l'achèvement du travail de développement, une fois que la qualité technique du système est démontrée. Elle permettra de garantir la logique et la complétude du système [Brès, 93].

7.1.1 .Application au commerce électronique

Afin de mettre en valeur notre travail, notre choix d'application est fait sur le commerce électronique, vu sa popularité et sa simplicité.

7.1.1.1 . Le cycle de vie d'une activité « achat électronique »

Dans cette section nous allons présenter des scénarios avec des diagrammes d'objets corresponds, décrivent le déroulement d'une activité d'achat par la technologie du commerce électronique, cette technologie sera aborder d'une manière détaillée dans l'annexe B.

Afin d'aboutir à un modèle qui décrit le déroulement de l'activité d'achat électronique, il faut instancier le méta modèle, ce dernier a été présenté par le diagramme de classe. L'instanciation est montrée par les diagrammes d'objets présentés dans la section suivante.

1) Le partage de l'information

Initialement le client consulte le catalogue des produits proposés par l'entreprise ainsi que les conditions et les contraintes de vente maintenues parle site.

Le diagramme suivant présente cette étape :

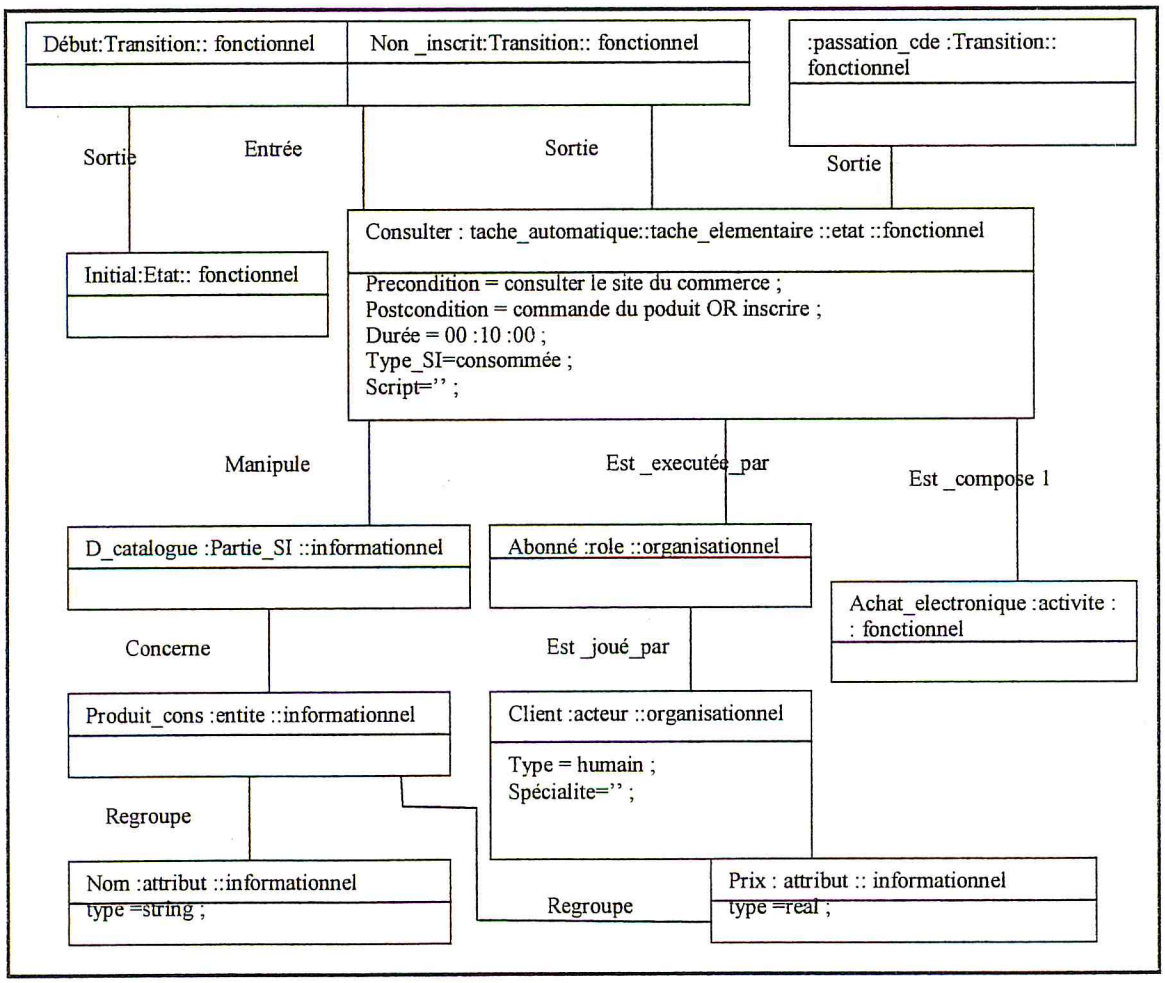


Figure III-50: Diagramme d'objet « consulter »

Afin de pouvoir participer au cycle d'achat, le client doit être abonné dans ce site autant que client. Il doit introduire ses cordonnées (nom, adresse, numéro de compte bancaire ...) auprès d'un formulaire qui lui sera fourni par le site. Tout client doit avoir un numéro qui le différencie des autres.

Bien sûr, la demande d'inscription sera vérifiée et examinée par une application appropriée, pour cela la demande d'inscription sera soit acceptée ou bien rejetée.

Le diagramme suivant illustre l'inscription du client :

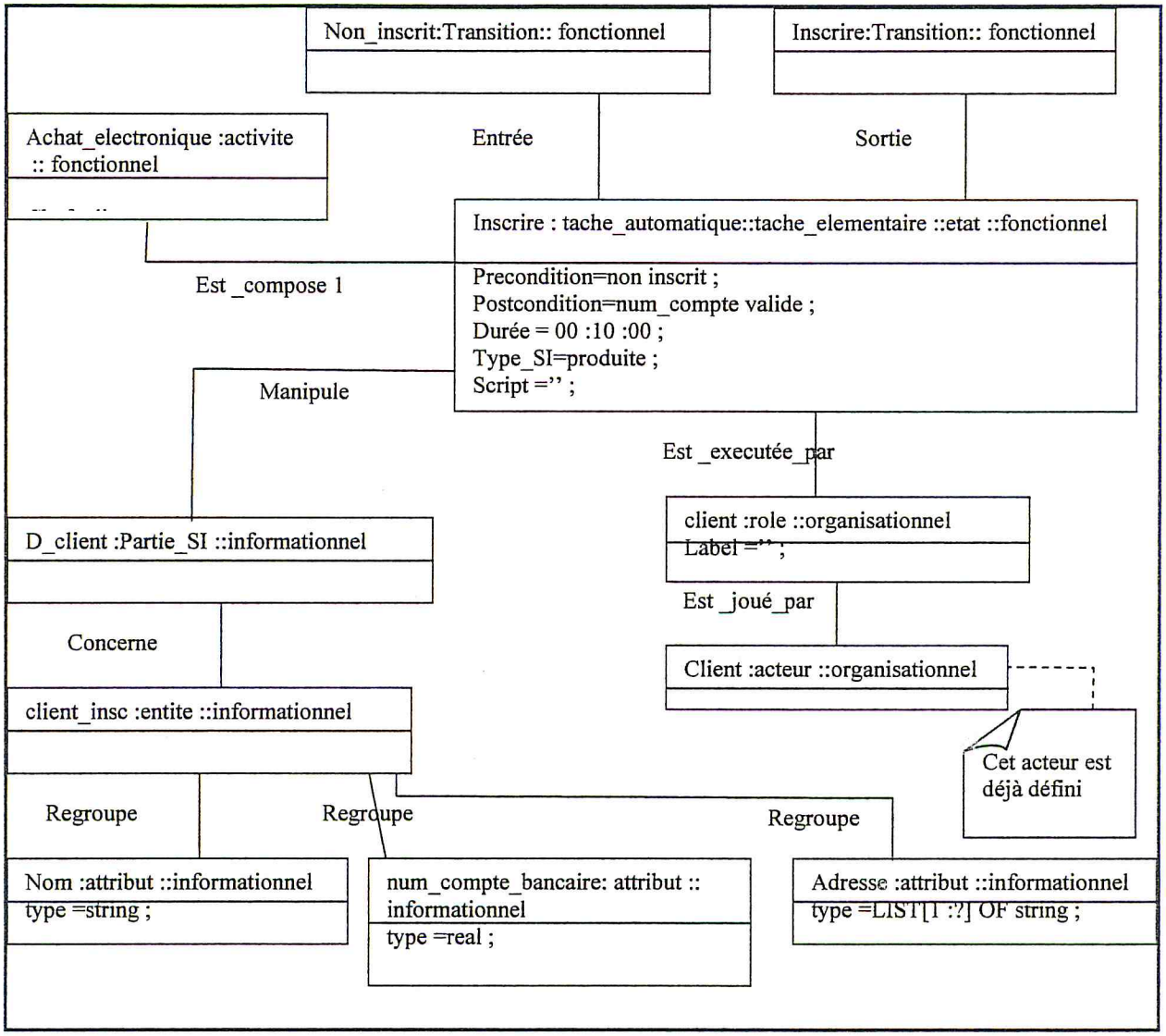


Figure III-51 : Diagramme d'objet « inscrire »

2) La commande

Une fois le client est inscrit, il peut effectuer une opération d'achat en remplissant une commande indiquant la liste des produits avec les éventuelles quantités. Comme le cas d'une demande d'inscription, la commande doit être vérifiée, afin de pouvoir décider son acceptation ou son rejet.

Le diagramme suivant présente cette étape :

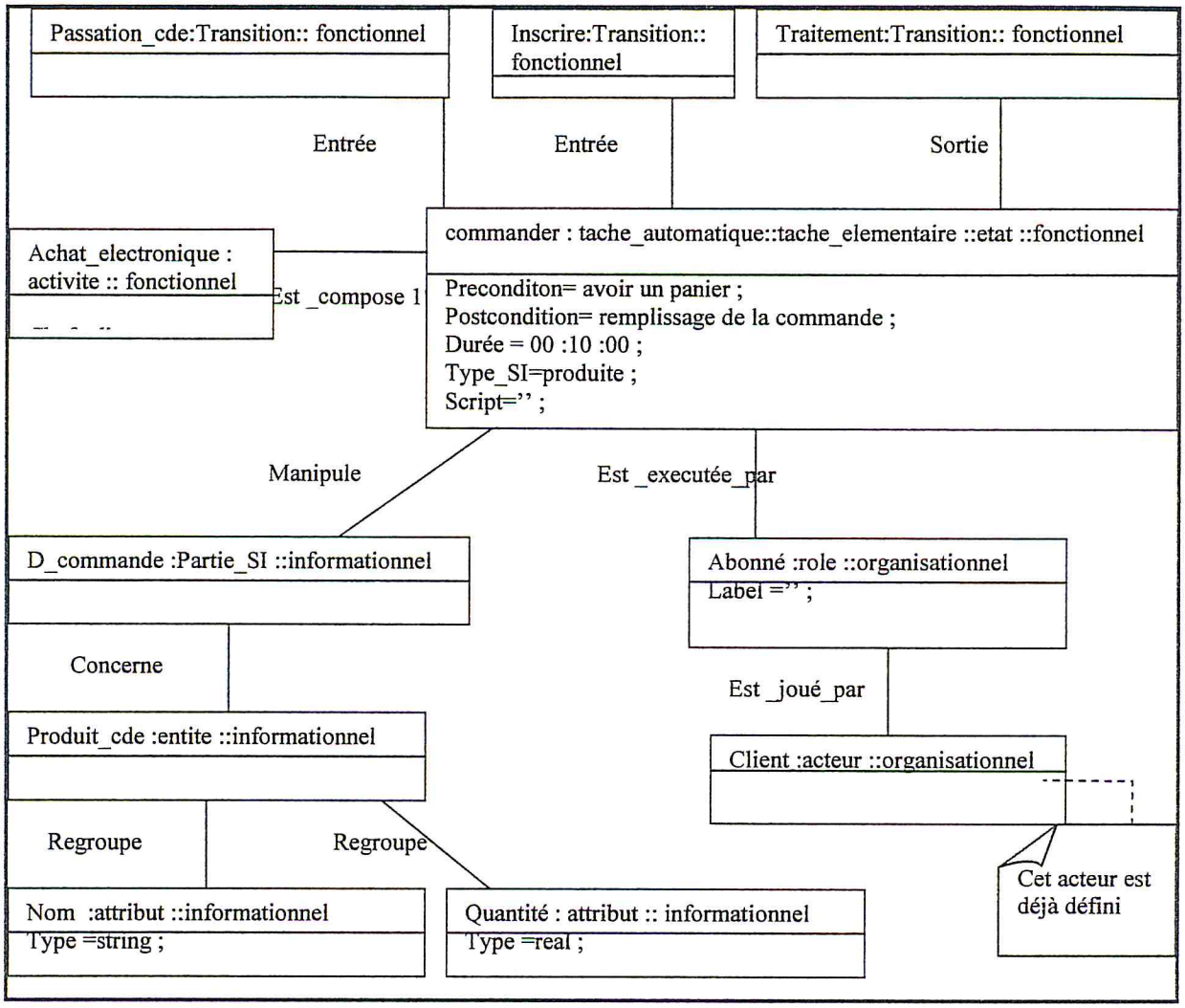


Figure III-52: Diagramme d'objet « commander »

La commande du client sera prise en considération par un agent commercial qui chargé d'effectuer le traitement de la commande. La Figure III-53 illustre cette opération.

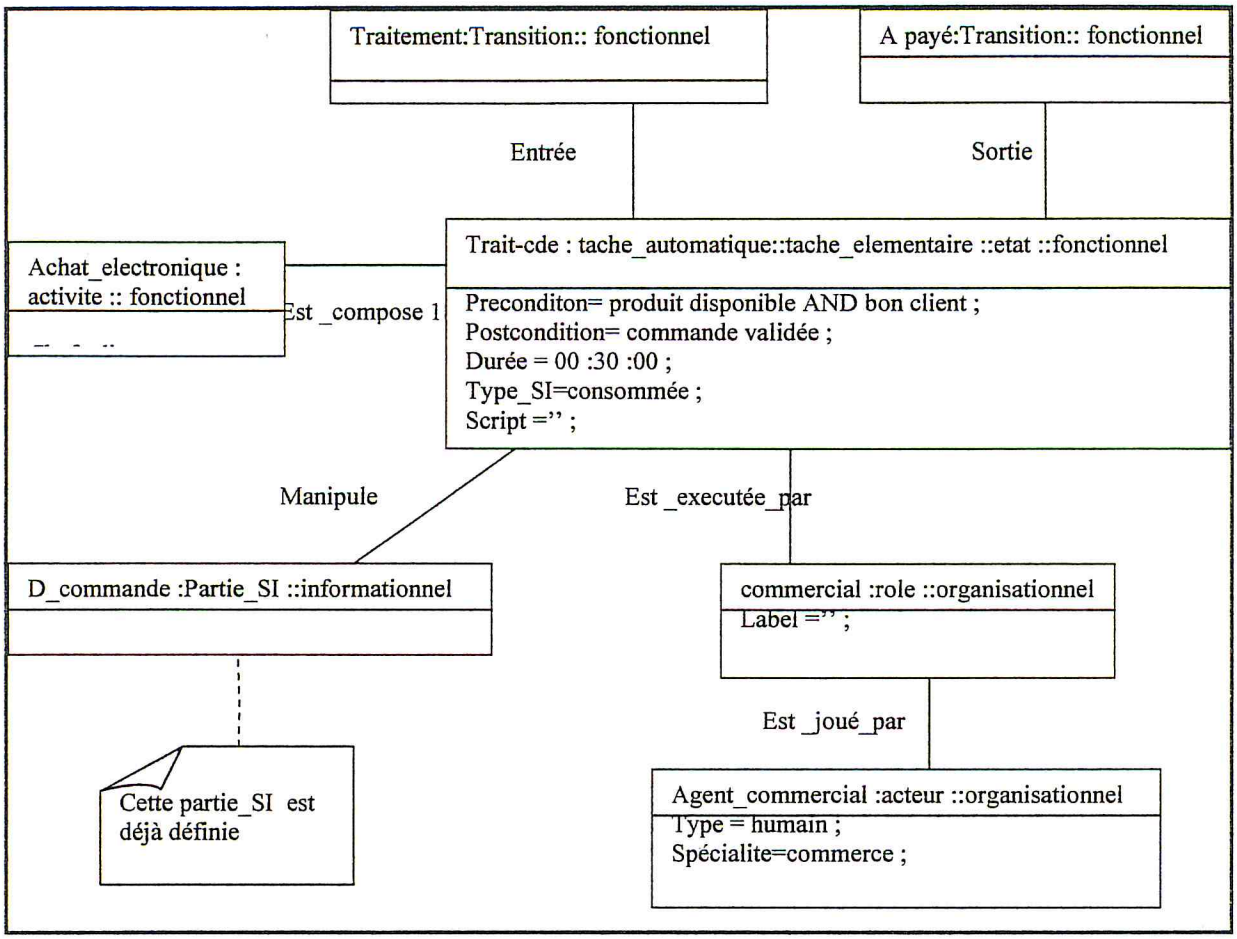


Figure III-53: Diagramme d'objet « traitement de la commande »

3) Le paiement

Après avoir accepté la commande, une facturation est enchaînée par la suite, elle consiste à remplir une facture, indiquant le montant globale des produits. La facture sera envoyée au concernée pour finaliser l'opération et effectuer la livraison des achats.

Le diagramme suivant présente cette étape :

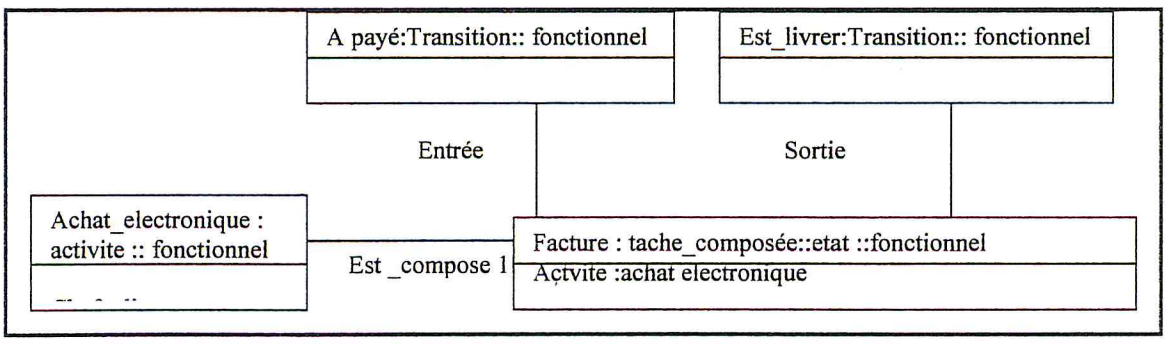


Figure III-54: Diagramme d'objet « facturer »

4) La livraison

Le service des ventes et facturation est chargé de la préparation du bon de livraison en deux exemplaires, un sera envoyé au client et l'autre sera envoyé au responsable du service livraison. Le client et le responsable de service de livraison peuvent maintenant procéder à la livraison.

Le diagramme suivant présente cette étape :

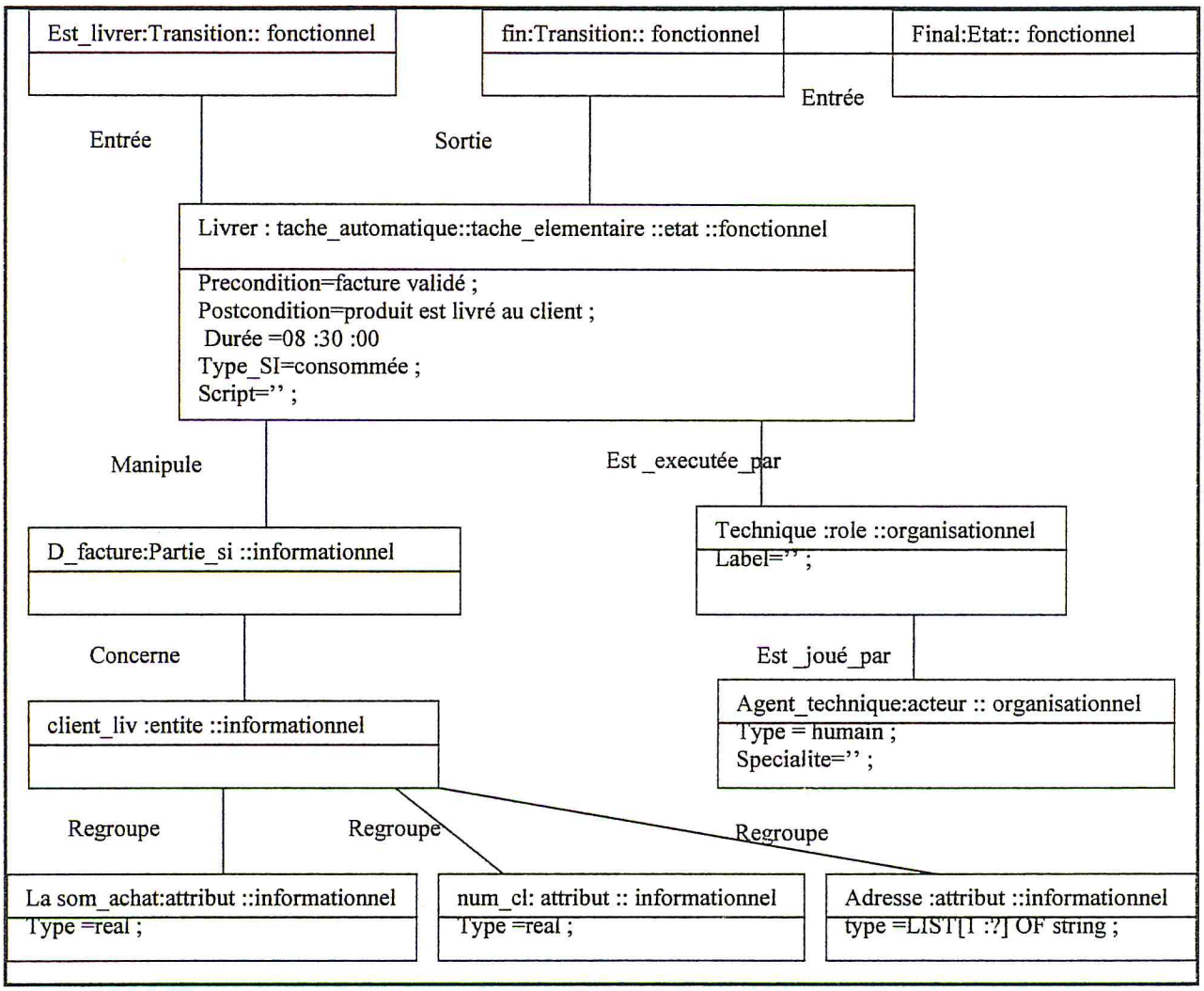
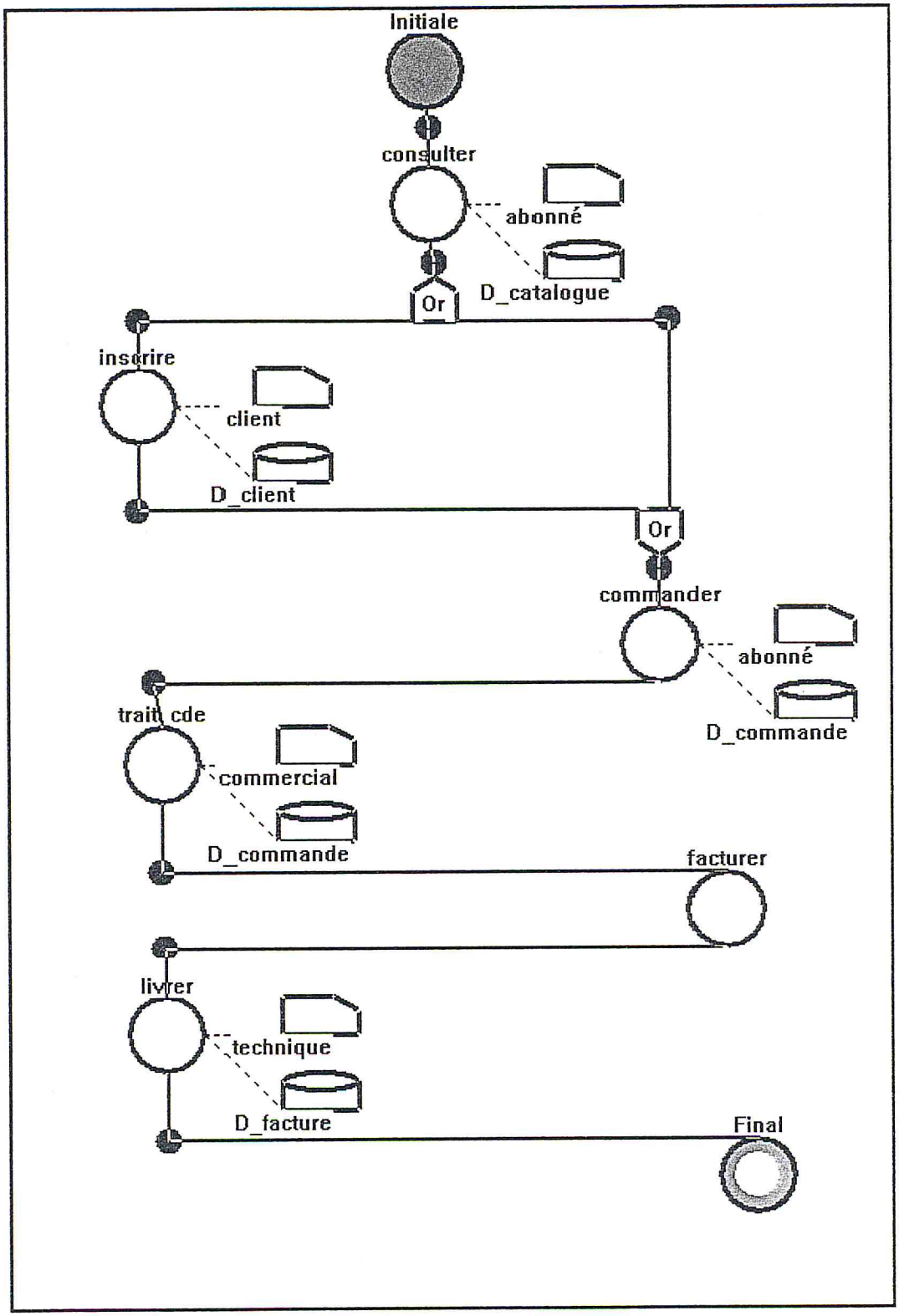


Figure III-55: Diagramme d'objet « livrer »

7.1.1.2 . Le modèle de processus

L'activité d'achat électronique, est constitué de cinqs tâches élémentaires : *consulter*, *inscrire*, *commander, trait_cde (traitement de la commande)* et *livrer*; et une tâche composée, *facturer*. Pour chaque tâche élémentaire on associe le rôle chargé de sa réalisation et l'ensemble de données manipulées par cette tâche. La tâche complexe *facturer* doit être décomposée en tâches élémentaires.

Le modèle suivant est une représentation graphique de l'activité d'achat électronique :



FigureIII-56 : Modèle de processus achat électronique d'abonnement

7.1.2 .Instanciation du schéma EXPRESS

Le fichier physique présenté dans la figure suivante est généré par l'outil Ecco Toolkit, il contient l'ensemble d'instance décrivant le modèle de processus présenté dans la figure III-57.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('fichier physique'),'2;1');
FILE_NAME('Workflow_process','2003-05-06T 1:20:04',
          ('Boumahdi Fareh'),
          ('UNIV-BLIDA'),
          'ECCO RUNTIME SYSTEM BUILT-IN PREPROCESSOR V3.0.5',
          'ECCO RUNTIME SYSTEM V3.0.5',
          'ADMINISTRATEUR');
FILE_SCHEMA(('ORGANISATION','WORKFLOW_PROCESSUS','PARTIE_SI'));
ENDSEC;
/* ISO 10303-21 file generated by ECCO Toolkit, PDTec GmbH, Germany */
DATA;
#1=TACHE_AUTOMATIQUE('consulter', 'achat-\S\ilectronique', 'consulter le
site du', 'commande du pod', '00 :10 :00 ', 'abonn\S\i', 'D_catalogue', '');
#2=TACHE_AUTOMATIQUE('inscrire', 'achat-\S\ilectronique', 'non inscrit ',
'num_compte valid', '00 :10 :00 ', 'client', 'D_client', '');
#3=TACHE_AUTOMATIQUE('commander', 'achat-\S\ilectronique', 'avoir un panier
', 'remplissage de la c', '00:10:00', 'abonn\S\i', 'D_commande', '');
#4=TACHE_AUTOMATIQUE('trait_cde', 'achat-\S\ilectronique', 'produit
disponible ', 'commande valid\S\ie', '00:30:00', 'commercial', 'D_commande',
'');
#5=TACHE_AUTOMATIQUE('livrer', 'achat-\S\ilectronique', 'facture valide',
'produit livr\S\i au clie', '8:30:00', 'technique', 'D_facture', '');
#6=TRANSITION('D\S\ibut', 'Initiale', 'consulter');
#7=TRANSITION('Transition', '', '');
#8=TRANSITION(' Non _inscrit', 'consulter', 'inscrire');
#9=TRANSITION('passation_cde', 'consulter', 'commander');
#10=TRANSITION('inscrire', 'commander', 'inscrire');
#11=TRANSITION('traitement', 'commander', 'trait_cde');
#12=TRANSITION('a payer', 'trait_cde', 'facturer');
#13=TRANSITION('est_livr\S\i', 'facturer', 'livrer');
#14=TRANSITION('finale', 'livrer', 'Final');
#15=TACHE_COMPOSE('facturer', 'achat-\S\ilectronique');
#16=ACTIVITE('achat-\S\ilectronique');
#17=ROLE('technique', $);
#18=ROLE('commercial', $);
#19=ROLE('client', $);
#20=ROLE('abonn\S\i', $);
#21=ACTEUR('client', $);
#22=ACTEUR('agent_commercial', $);
#23=ACTEUR('agent_technique', $);
#24=SI('D_facture');
#25=SI('D_commande');
#26=SI('D_client');
#27=SI('D_catalogue');
#28=ENTITE('produit-cons', 'D_catalogue');
#29=ENTITE('client_insc', 'D_client');
#30=ENTITE('produit-cde', 'D_commande');
#31=ENTITE('client_liv', 'D_facture');
#32=ATTRIBUT('Nom', 'string', 'produit-cons', 'D_catalogue');
#33=ATTRIBUT('Prix', 'real', 'produit-cons', 'D_catalogue');
#34=ATTRIBUT('Nom', 'string', 'client_insc', 'D_client');
#35=ATTRIBUT('NUM_compte_bancaire', 'real', 'client_insc', 'D_client');
#36=ATTRIBUT('Adresse', 'LIST[1 :?] OF string ', 'client_insc',
'D_client');
ENDSEC;END-ISO-10303-21;
```

Figure III-57 : Une partie du fichier physique de processus achat électronique.

7.2 . Validation des cas d'utilisation

Dans cette section des diagrammes de séquences sont présentés avec des copies d'écran pour illustrer l'activité de validation et pour être sur que les fonctionnalités du systèmes décrites dans les cas d'utilisation sont réalisées réellement par notre outil.

Dans ce qui suit nous présentons quelque cas d'utilisation et l'annexe D illustre les autres cas.

7.2.1 .Définition de l'organisation

Pour définir les rôles qui participent à l'accomplissement des tâches, il faut définir tous les acteurs qui jouent le rôle modélisé (Figure III-58).

Comme nous montre la Figure III-59, lors de la définition des rôles, une boite de dialogue s'affiche. Elle contient tous les acteurs définis, et l'utilisateur désigne les acteurs qui jouent le rôle approprié.

7.2.1.1 . Validation du cas définition des acteurs

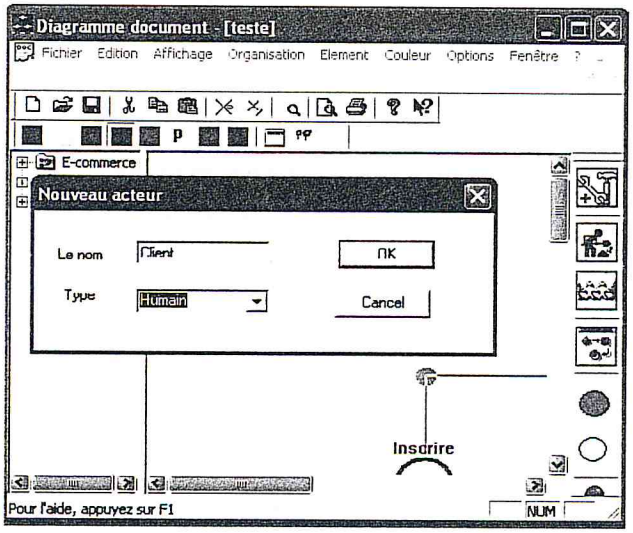


Figure III-58: Définition des acteurs

7.2.1.2 . Validation du cas définition des rôles

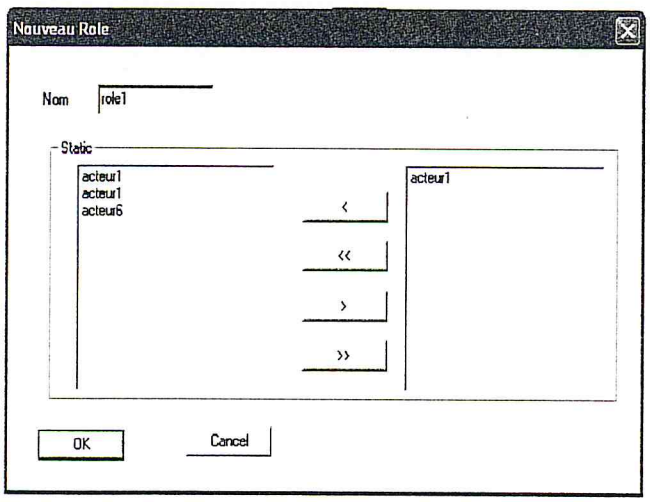


Figure III-59: Définition des rôles

7.2.2 .Définition des données

7.2.2.1 . Définition des parties SI

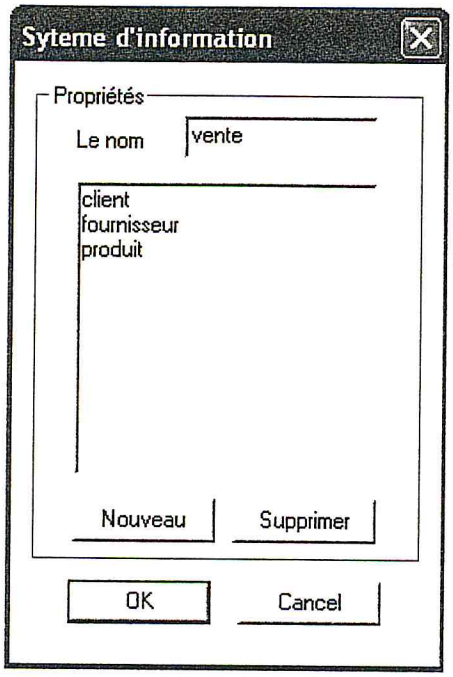


Figure III-60: Définition de SI

7.2.3 .Définition du fonctionnement du processus

7.2.3.1 . Validation du cas définition des tâches

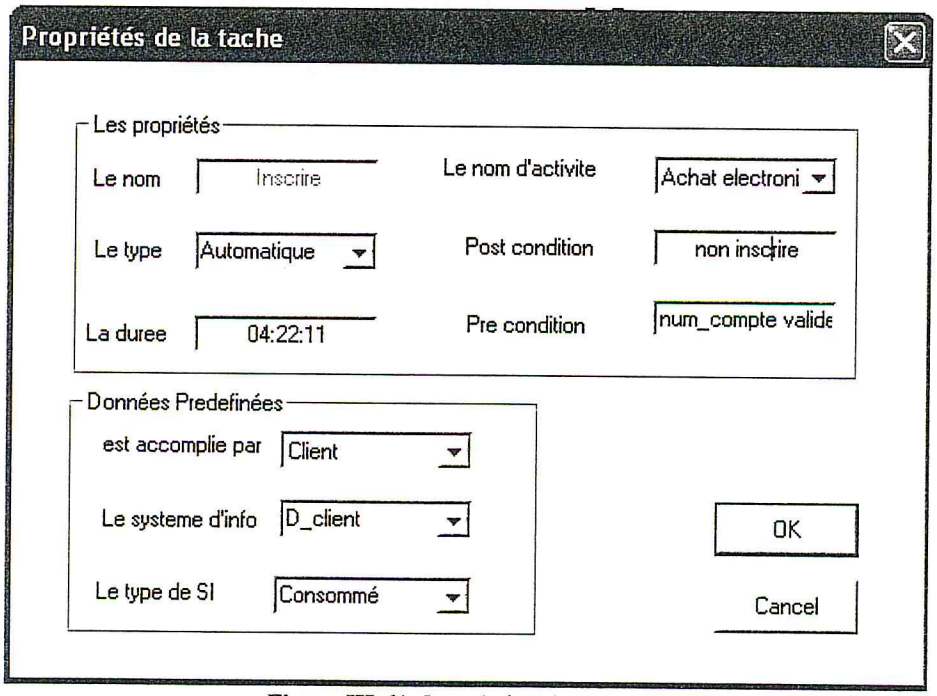


Figure III-61: La création d'une tâche

8. Conclusion

Dans ce présent chapitre nous avons présenté, les étapes de développement de notre outil, en utilisant une démarche dans un environnement orienté objet à savoir la modélisation et la conception. Les logiciels objet sont plus extensibles, plus proches des utilisateurs, plus faciles à maintenir, mais ils restent toujours difficiles à réaliser. Dans le développement de notre outil le modèle en cascade représente la démarche suivie, et l'utilisation de la notation UML montre l'environnement.

Pour développer notre outil nous avons commencé par la spécification des besoins, qui décrit les fonctionnalités du système, présentés par les diagrammes des cas d'utilisation, séquence, collaboration, et d'activités. Puis nous avons entamé l'étape d'analyse pour déterminer les éléments intervenants dans notre système.

La troisième étape est la phase la plus importante dans le développement de l'outil, elle commence par une conception globale qui décrit l'architecture du système suivie par une autre détallée. L'étape qui se suit à pour but d'implémenter notre solution. Quand a la dernière étape, une validation est présenté après un test sur le E-commerce.

L'outil de définition de workflow que nous avons développé, respecte les principales règles ergonomiques des programmes fonctionnant sous Windows. Nous avons conçu notre outil pour permettre une prise en main facile par l'utilisateur.

Notre outil n'est pas réservé à des experts pertinemment formés, mais bien un outil utilisable par tous les participants d'un projet d'organisation. Il assure la modélisation des processus et la génération des définitions, qui sont interprétables par un moteur de workflow.

Conclusion générale

L'objectif primordial de ce mémoire consiste à analyser et concevoir un outil de définition de workflow, pour permettre la modélisation des processus workflow et la génération automatique des définitions associées à ces processus.

La présence d'un processus de développement formalisé, bien défini et bien géré est un facteur de réussite d'un projet. Pour cette raison nous avons suivi le modèle en cascade, utilisant UML comme un langage de modélisation.

Au cours de notre projet nous avons développé un outil de définition, entièrement dédié aux opérationnels en charge de la définition des processus dans l'entreprise. Il permet grâce à son ergonomie, à un utilisateur sans connaissance technique de s'approprier son environnement et de modéliser ses processus, et tout cela sans l'assistance du service informatique. Cet outil est simple, accessible et convivial d'utilisation, permettant aux usagers de représenter graphiquement leur processus métier, leurs activités, leurs enchaînements et les règles associées, dans un formalisme dépend du méta modèle correspondant au type d'activité modélisé. Ce méta modèle regroupe les composants processus, organisation et SI. Nous avons interprété également le méta modèle par le langage de spécification EXPRESS aboutissant à un schéma d'implémentation en utilisant l'environnement Ecco Toolkit, il s'agit d'un contrôleur de modèle EXPRESS.

Le système ainsi développé, offre donc de nombreux avantages aux utilisateurs notamment la possibilité de modéliser leur processus puis de générer automatiquement des populations d'instances correspondantes.

Ce projet très enrichissant nous a donné l'occasion de nous plonger dans un problème concret, ce qui nous a permis d'affiner les connaissances acquises lors de notre projet. Nous avons eu la chance d'être intégré à un projet important impliquant un grand nombre de personnes et faisant appel à des technologies novatrices.

Le début a été difficile, ce qui est principalement dû à la multiplicité des concepts EXPRESS et des notations UML. Cependant, ces difficultés nous ont permis d'acquérir un esprit de synthèse, ce qui est loin d'être négligeable.

Ce projet nous a aussi permis de progresser en terme de rigueur, de méthodologie et aussi bien concernant la conduite de projet que la programmation. Le projet nous a de plus permis

d'approfondir nos connaissances sur le workflow, sur le langage EXPRESS et UML, sur le langage visuel C++, ainsi que l'acquisition de connaissance sur des techniques récentes de modélisation de données et tout particulièrement la méta modélisation.

Ce travail a révélé un certain nombre de perspectives qui sont résumées en :

La première concerne le développement de notre outil sous un environnement Intranet, ce développement permet à plusieurs utilisateurs de modéliser le même processus.

Une autre perspective réalisable à long terme, qui dépasse le cadre de notre projet est l'interprétation de la définition générée par un moteur de workflow, ce dernier chargé d'affecter les tâches aux personnes responsables à leurs réalisations.

Bibliographie

[Ait Ameur, 02] Cours EXPRESS 2002, ENSMA LISI disponible au site : http://www.lisi.ensma.fr

[Bernardi, 02] F. Bernardi, « Méthode d'analyse orienté objet UML », Dunod, 2002

[Blau, 01] Marc Blau, « Gestion électronique documentaire systèmes de gestion de workflow et possibilités d'intégration de ces systèmes », diplôme postgrade en informatique et organisation, université de Lausanne, soutenue 2001-2002.
Disponible à l'adresse : http://inforge.unil.ch.

[Bres, 93] J.Bres, « ateliers de genie logiciel », Masson, 1993.

[Dehaninsala, 02] Hondjack DEHANINSALA, « Analyse et Implémentation d'un modèle objet EXPRESS dans une base de donnée relationnelle objet : PostgreSQL », mémoire d'ingéniorat de l'université de IAI, 2001/2002.

[Ecco, 01] Ecco Toolkit. Programmer's Guide. PDTec. Version 3.0.0 Date 2001-12-10

[Fannader &al 00] Rémy Fannader, Hervé Lerroux, « UML principes de modélisation », Dunod, 2000.

[Francis, 02] FRANCIS LORENTZ, « Commerce électronique une nouvelle donnée pour les consommateurs, les entreprises, les citoyens et les pouvoirs publics, disponible à l'adresse :
http://www.finances.gouv.fr/commerce_electronique.htm

[Frey, 99] FREY, « La gestion de documents à travers une application Workflow », diplôme postgrade en informatique et organisation, université de Lausanne, soutenue 1999-2000.
Disponible à l'adresse : http://inforge.unil.ch.

[Gabay, 98] Joseph Gabay, « Merise vers OMT et UML », Masson, 1998

[Gardarin, 02] Georges Gardarin, « BASE de données », Eyrolles, paris 2002

[Hamdah, 03] Hamdah Mohamed, « modélisation de workflow dans un environnement orienté objet », mémoire de magistèr USTHB, 2002/2003.

[Hammouda & al, 02] Hammouda. M, Ouchen O « modélisation d'une application de commerce électronique par les techniques de workflow », mémoire d'ingéniorat, USTHB, 2001/2002

[kettani & al, 01] kettani N, Mignit D.Paré P.,Rosenthal-Sabroux C, « De Merise à UMl », Eyrolles, Paris,2001

[khochafiane& al, 98] S.khochafiane, M. Buckiewiez, « Groupware et Workflow », Masson, 1998

[Levan, 99a] Serge K. Levan, « Le projet Groupware », Eyrolles, 1999

[Levan, 99b] Serge K. Levan, « Le projet workflow », Eyrolles, 1999

[Muller, 97] Pierre-Alain Muller, « Modélisation objet avec UML », Eyrolles, 1997

[Muller, 01] Pierre-Alain Muller, « Modélisation objet avec UML », Eyrolles, 2001

[Paschalidès, 01] Michel Paschalidès, « Le Commerce électronique », disponible à l'adresse :http://sawww.epfl.ch/SIC/SA/publications/FI98/fi-2-98/2-98-page1.html

[Pillou, 02] Jean-François Pillou, « Introduction à la programmation orienté objet », 2002 disponible à l'adresse :
http://www.commentcamarche.net/poo/poointro.php3.

[Saadoun, 96] Mélissa Saadoun, « Le projet groupware », Eyrolles, 1996

[Saadoun, 02] Mélissa Saadoun « Le Workflow Pour automatiser les procédures et tâches répétitives », INEDIT, 2002

[Sardet, 99] Eric Sardet, « Intégration des approches modélisation conceptuelle et structuration documentaire pour la saisie, la représentation, l'échange et l'exploitation d'informations. Application aux catalogues de composants industriels », doctorat de l'université de POITIERS, Soutenue le 4 Octobre 1999.

[Spidy, 94] Philip SPIDY et Doug SCHENK « ISO 10303-11 part11 : EXPRESS Language Refenece Manual » Novembre 1994

[Sommerville, 88] Lan Sommerville, « Le génie logiciel et ses applications », paris, 1988.

[Wfmc, 95] Workflow management coalition, the reference model, document number TC 00-1003, 1995 disponible à l'adresse : www.wfmc.com.

ANNEXE A.
LA SPECIFICATION DU META MODELE

```
SCHEMA Workflow_process;

USE FROM organisation (role);
USE FROM Donnees_manipulees(PARTIE_SI);

ENTITY processus;
      Nom_processus : t_nom;
      Activites :LIST OF activite;
UNIQUE
      Ur :nom_processus;
END_ENTITY;--processus

TYPE t_nom = STRING;
      WHERE LENGTH(SELF)>0;
END_TYPE;

ENTITY activite;
      nom_activite : t_nom;
      etats : LIST OF etat;
INVERSE
      Compose : S [1:?] OF processus FOR activite;
UNIQUE
      Ur : nom_activite;
END_ENTITY; --activite


ENTITY etat;--tâche
ABSTRACT SUPERTYPE  OF (ONEOF(tache_elementaire, tache_composee));
      nom_etat : t_nom;
      transition_entree    :     SET [0:?] OF transition;
      transition_sortie    :     SET [0:?] OF transition;

INVERSE
      compose_1 : activite FOR etats;
      compose_2 : tache_composee FOR etatss ;


END_ENTITY; --etat
```

```
ENTITY transition;
      nom_transition     : t_nom;
INVERSE
      etat_entree    : SET [0:?] OF etat FOR transition_sortie;
      etat_sortie    : SET [0:?] OF etat FOR transition_entree;
UNIQUE
      Ur :nom_transition;
END_ENTITY ;--transition

ENTITY tache_composee
SUBTYPE OF (etat);
      nom_tache_composee : t_nom;
      etatss : LIST [1 :?]OF etat;
UNIQUE
      Ur :nom_tache_composee;

END_ENTITY; --tache_composee

ENTITY tache_elementaire
ABSTRACT SUPERTYPE OF (ONEOF (tache_automatique, tache_manuelle ))
SUBTYPE OF  (etat);
      nom_tache_elementaire :t-nom ;
      Pre condition    : STRING ;
      Post condition   : STRING ;
      Est_accomplie_par : role;
      *l''entité role est référencée à partir le schéma organisation via le
      mot clés USE FROM *)
      Dates : LIST [1 :4] OPTIONNEL t_date;
      Duree :REAL;
      Donnees : partie_SI; --l'entité PARTIE_SI est Référencée
      donnees_manipulees)par USE FROM
      traitement :t_traitement ;

UNIQUE
      ur :nom_tache_elementaire ;
WHERE
      wr1 : duree > 0 ;
      wr2 :SIEZOFEOF(QUERY(x <* est_accomplie_par))<> 0;
END_ENTITY ; --tache_elementaire

TYPE t_traitement =ENUMERATION OF(consomme, produit, echange);

END_TYPE;

ENTITY tache_automatique
SUBTYPE OF (tache_elementaire)
Script :t_script ;

END_ENTITY;-- tache_automatique

ENTITY tache_manuelle
SUBTYPE OF (tache_elementaire) ;
Description :t_description ;

END_ENTITY;-- tache_manuelle

RULE autant_tache_que_role FOR (tache_elementaire,role);
WHERE
```

```
      WR: SIZEOF (tache_elementaire)>= SIZEOF(role) ;
END_RULE;--autant_tache_que_role

END_SCHEMA; --workflow_processus

SCHEMA  organisation;

USE FROM workflow_processus (tache_elementaire);

ENTITY role
ABSTRACT SUPERTYPE OF (ONEOF(role_individuel, role_groupe));
      nom_role  : t_nom;
      est_joue_par  : SET[1:?] OF Acteur;
      chef :acteur ;
      droit_acces :t_droit;
INVERSE
      Accomplie : SET[1:?] OF tache_elementaire FOR est_accomplie_par;

UNIQUE
      Ur : nom_role;

END_ENTITY; -- role

TYPE t_droit=ENUMERATION OF (lecture, ecriture, execution);

END_TYPE;

ENTITY role_individuel
SUBTYPE OF(role);
      Joue_dans : OPTIONNEL role;
END_ENTITY;-- role_individuel

ENTITY role_groupe
SUBTYPE OF(role);
      nbr_acteur : NUMBER;
INVERSE
      Sous_role : role_individuel FOR joue_dans;
END_ENTITY;

ENTITY acteur
ABSTRACT SUPERTYPE OF ((ONEOF (humain, outil)));
      nom_acteur : STRING;
INVERSE
      joue      : SET [1 :?] OF role FOR est_joue_par;
      est_chef  : role FOR chef;
UNIQUE
      Ur : nom_acteur ;
END_ENTITY; --acteur

ENTITY humain
SUBTYPE OF (acteur) ;
      Specialite :STRING;
      CV : t_CV;
      Poste : t_poste;
      Adresse :t_adresse;--adresse électronique ou physique
END_ENTITY ;--humain

ENTITY t_cv;
      Diplopmes : LIST[0:?] OF STRING;
      date_recrutement :t_date;
```

```
END_ENTITY; -- t_cv

ENTITY outil
SUBTYPE OF (acteur) ;
      Version     : t_version;
      Plate_forme :t_plate_forme;
      Companie : t_compagnie;
      endroit : t_endroit;
UNIQUE
      Ur : endroit;
END_ENTITY; --outil

TYPE t_outil =ENUMERATION OF (WORD, EXCEL);
END_TYPE; --t_outil

END_SCHEMA; --organisation
SCHEMA donnees_manipulees;

USE FROM Workflow_processus(tache_elementaire);

ENTITY Partie_SI;
      non_systeme : t_nom;
      entites     :LIST [1 :?]OF entite;

INVERSE
      est_manipule_par : SET [1:?] OF tache_elementaire FOR manipule;

END_ENTITY ;-- Partie SI

ENTITY entite;
      nom_entite :t_nom;
      super_classe : SET [0:?] OF entite;
      attributs  :LIST [1:?] OF attribut;
INVERSE
      sous_classe : SET[0 :?] entite FOR super_classe;

END_ENTITY;

ENTITY attribut;
      nom_attribut  : t_nom;
      type_attribut : t_attribut;
END_ENTITY;

TYPE t_attribut= SELECT( simple , collection, utilisateur, Defini);

END_TYPE;

TYPE simple = SELECT (INTEGER, STRING, REAL, BOOLEAN, NUMBER, LOGICAL);
END_TYPE;

TYPE collection = SELECT(entite, t_attribut);
END_TYPE;

TYPE utilisateur = SELECT(collecton, entite);
END_TYPE;

TYPE defini=SELECT(entite, t_attribut);
END_TYPE;

END_SCHEMA;
```

ANNEXE B. LE COMMERCELECTRONIQUE

A. Définitions

Le commerce électronique est définit par « l'ensemble entier des processus qui supportent des activités commerciales sur un réseau » [Francis, 02]. Ces activités traitent les renseignements concernant les produits et les évènement de l'exposition, les services, les fournisseurs, les consommateurs, les annonceurs, la sécurité des transactions »...[Hammouda&a] 02].

B. Le développement de commerce

Du point de vue commercial, Internet se présente comme le successeur logique de l'évolution du marché. Le commerce n'a cessé de s'étendre. De la société de production locale (à exploitation directe) à une société de service; du petit marché local, aux commerces du centre ville, puis aux grands magasins, puis à la surface commerciale en dehors des zones habitables, et enfin à l'achat par correspondance. Et quand la situation géopolitique l'a permis, la mondialisation et la globalisation du marché ont confirmé Internet comme le mode marchand de l'avenir [Francis, 02]. Du point de vue du consommateur, rien n'est plus simple qu'acheter depuis chez soi, n'importe quand et n'importe où dans le monde. Du point de vue du vendeur, les intérêts sont multiples:

- vaste clientèle potentielle non limitée géographiquement;
- coût de diffusion des informations réduit;
- frais d'exploitation réduits, concentration des lieux de stockage dans des zones décentralisées;
- suppression éventuelle des intermédiaires entre producteur et consommateur.

Du point de vue, consommateur les intérêts sont aussi multiples:

- Une offre mondiale et un choix infini auquel vous avez accès à partir de votre place.
- Des horaires d'ouverture 24 h sur 24.
- Des produits livrés directement à domicile.

Il existe deux stratégies de politique de marketing proposées sur le net. L'approche directe du consommateur via des catalogues électroniques à libre accès sur Internet, et celle d'entreprise à entreprise via des catalogues électroniques destinés uniquement aux collaborateurs via un réseau Intranet.

C. Comparaison entre le commerce électronique et le commerce traditionnel

En comparant le cycle de vente d'une transaction traditionnelle à celui d'une transaction électronique, on peut noter bien des similitudes. Seules les méthodes d'obtention et de transmission de l'information variant.

Dans une transaction traditionnelle, de multiples vecteurs de communication sont indispensables. Cette diversité a pour conséquence de compliquer la coordination des opérations et d'allonger considérablement le temps de traitement d'une commande. En revanche, dans le cas d'une transaction électronique, l'information est numérisée de bout et il n'existe qu'un seul vecteur de communication : le Web.

Avant, l'acheteur faisait une demande mais il passe à la commande qu'après avoir obtenu l'accord, choisi le fournisseur chez qui le produit est disponible en se basant sur des formulaires imprimés ou des lettres ; tandis que le commerce électronique permet à l'acheteur de choisir le produit à partir d'un site Web, exploiter les techniques électroniques pour effectuer sa demande et la suivre, en utilisant un unique moyen de communication **(Web)** [Hammouda & al, 02].

L'achat sur Internet présente de grandes différences avec un achat dans un supermarché ou une boutique. Il se distingue également d'une commande dans un catalogue.

Etape de cycle de vente	**Commerce traditionnel (multiples vecteurs de communication)**	**Commerce électronique (unique vecteur de communication)**
Recherche d'informations Sur un produit.	Magazines, représentants, Catalogues	Page WEB
Commande de produit	Lettres, formulaires	E-mail
Confirmation de commande	Lettres, formulaires	E-mail
Vérification de prix	Catalogues	Catalogues en ligne
Vérification de disponibilité	Téléphone, fax	Catalogues en ligne
Passation de la commande	Formulaire imprimé	E-mail, page Web
Envoi/réception de commande	Fax, courier	E-mail
Vérification de disponibilité au dépôt	Formulaire imprimé, fax, téléphone	Base de données en line, page Web
Planification de la livraison.	Formulaire imprimé	Base de données en line
Génération de la facture	Formulaire imprimé	Base de données en line
Echéance de paiement	Formulaire imprimé	Base de données en line

Tableau B-1: Nouvelle et ancienne méthodes d'achat d'un produit [Hammouda & al, 02]

D. Les services de e-commerce

Le commerce électronique ne se limite pas à l'achat et à la vente de produits ou de services en ligne. Il comprend aussi l'utilisation d'Internet en tant qu'élément clé de la stratégie et des procédés d'une entreprise.

Pour tirer avantage du commerce électronique, une entreprise doit y intégrer tous ses procédés. Les entreprises qui se sont mises en réseau peuvent créer des produits, recevoir des commandes, communiquer avec les fournisseurs, organiser la production, gérer les livraisons et servir leur clientèle sans délais [Francis, 02]. Ainsi, elles réduisent leurs coûts de production, diminuent leur inventaire et améliorent la qualité du service et la satisfaction de la clientèle.

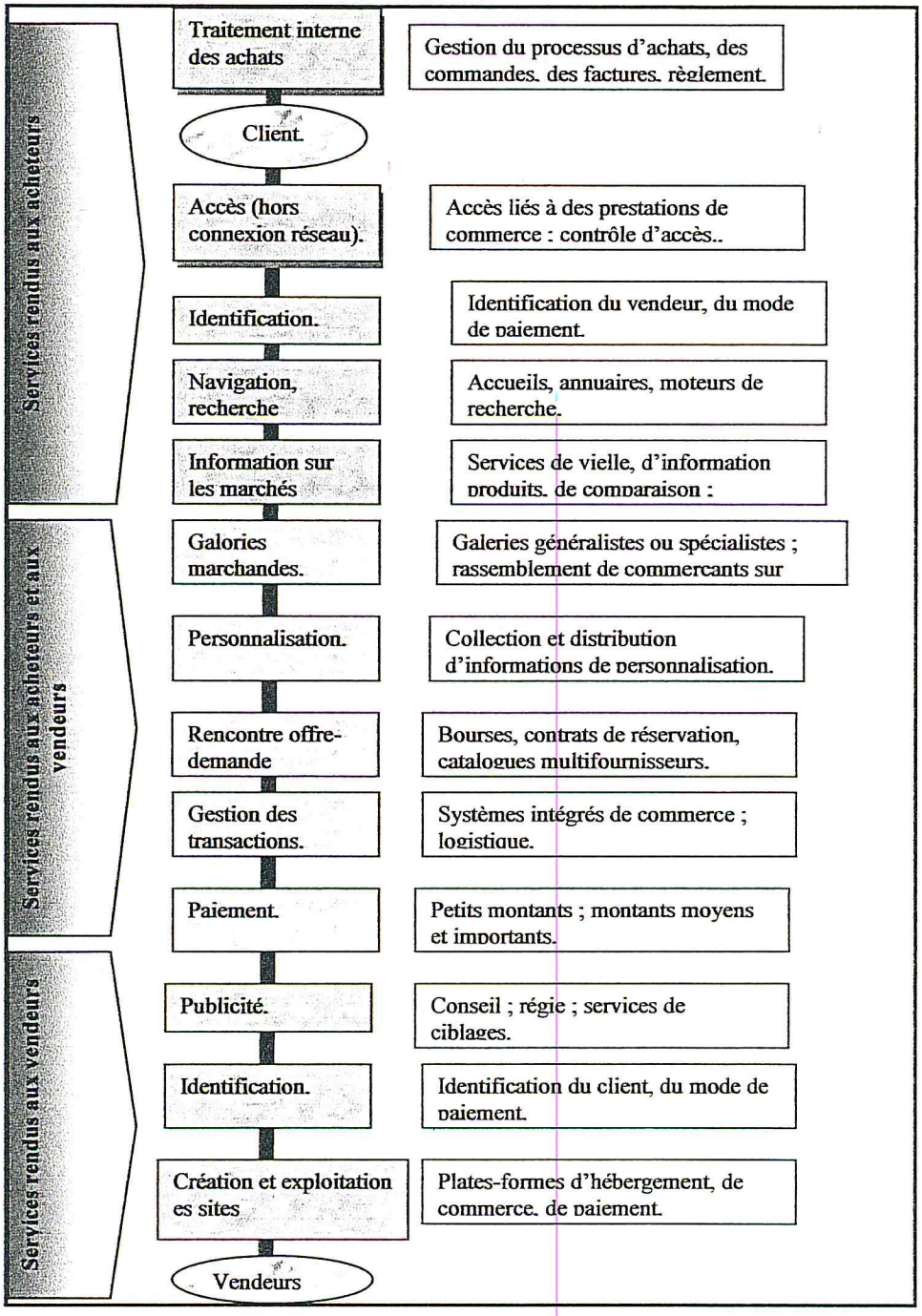


Figure B-1 : Les nouveaux services de E-commerce [Francis, 02]

E. Le cycle de vie d'une activité commerce électronique

Une activité de commerce électronique se base sur quatre familles de processus :

E.1 . Partage de l'information

Il faut d'abord concevoir un site qui permettra aux clients éventuels de trouver les produits et les services de l'entreprise. Le client parcourra le catalogue électronique et choisira des articles.

Il faut aussi rester à l'écoute des clients, en tant que communauté de réseau, constituent un excellent support de distribution des informations (par exemple proposer des pages de réponses aux questions les plus fréquemment posées).

E.2 . La commande

Le client doit au préalable inscrire sur le site, sinon il ne pourra pas participer au cycle d'achat.

Une fois que le client aura fini de faire son choix des produits, il trouvera un formulaire de commande en ligne. Habituellement, il doit inscrire son nom, son adresse, ...etc.

E.3 . Le paiement

Le calcul du total et des taxes, et la prise en compte des renseignements relatifs à l'expédition (qui est habituellement basée sur le lieu de livraison), du numéro de carte de crédit, de la date d'expiration et de l'adresse d'expédition se font ensuite à l'aide d'un logiciel de traitement des commandes [Francis, 02].

Le paiement ce fait via multiples moyens (carte crédit, chèque électronique, monnaie électronique,...).

Les systèmes de commerce électronique doivent être protégés contre toute opération de fraude [Paschalidès, 01]. Les services de sécurité doivent garantir, entre autres, l'authentification des données et des entités qui accèdent au système, le control d'accès, la confidentialité, et l'intégrité des données.

Les systèmes de sécurité se basent essentiellement sur la cryptographie qui permet le codage et le décodage des données [Paschalidès, 01]. Ces systèmes se basent soit sur une clé symétrique qui est partagée et gardée secrète entre l'émetteur et le récepteur, soit sur une clé publique et dans ce cas l'émetteur et le récepteur ne partagent pas la même clé : chaque personne possède deux clés, une rendue publique et l'autre gardée secrète.

E.4 . La livraison

- Soit le site d'achat groupé est le seul interlocuteur de client : il lui livre le(s) produit(s) et lui adresse la facture. Pour ce qui concerne la garantie, le client référé aux conditions générales du site.

- Soit le fournisseur livre le produit au client, lui adresse la facture et sera leur seul contact pour tout problème ultérieur (retour de marchandise, garantie).

F. Avantages et gains associés au commerce électronique

Le commerce électronique peut faciliter les échanges entre services internes, améliorer les relations fournisseurs clients et supprimer les contraintes de lieu et de temps, le tableau suivant résume les avantages de e commerce :

Fonctionnalité	Solutions électroniques	Avantages et gains
Catalogue de produits	*Catalogue électronique * Accès à distance	* réduction des coûts par suppression des coûts d'impression et de diffusion * facilité de mise à jour et diffusion instantanée * élargissement de la clientèle.
Diffusion marketing	* Site Web	* faire connaître sa société * élargir la cible de clientèle pour vendre au niveau mondial
Achat, recherche d'offre et de prix	*Achats par Internet	* élargir la sphère des achats sur un plan international, avec possibilité de recherche d'offres à prix plus compétitifs * augmenter les comparaisons de prix
Prise de commande	* Par site Web	* commande prise en compte plus rapidement permet de raccourcir les délais et d'éviter les ressaisies

Tableau B-2: Les avantages de E-commerce [Paschalidès, 01]

ANNEXE C. ECCO TOOLKIT

A. Historique

Bien que le langage EXPRESS ait été disponible pour plusieurs années, les outils les plus commerciaux sont limités pour vérifier la syntaxe et le convenance de la construction de la structure des données. Les proportions algorithmiques qui sont utilisées pour la spécification de contraintes de l'intégrité d'une manière prédominante sont insuffisamment maniées à tout ou pas.

Au RPK[16], ECCO Toolkit était développé [Ecco, 01] pour réduire l'effort en besoin de développer l'application, dont les modèles conceptuels sont spécifiés radicalement dans EXPRESS. ECCO est un environnement du développement intégré qui vérifie la convenance syntactique d'un modèle du données d'EXPRESS et traduire la spécification à une application exécutable ou bibliothèque qui peuvent être liées à une application externe existante. Depuis 1999, ECCO Toolkit est disponible de PDTec GmbH, karlsruhe, comme un produit du logiciel commercial [Ecco, 01].

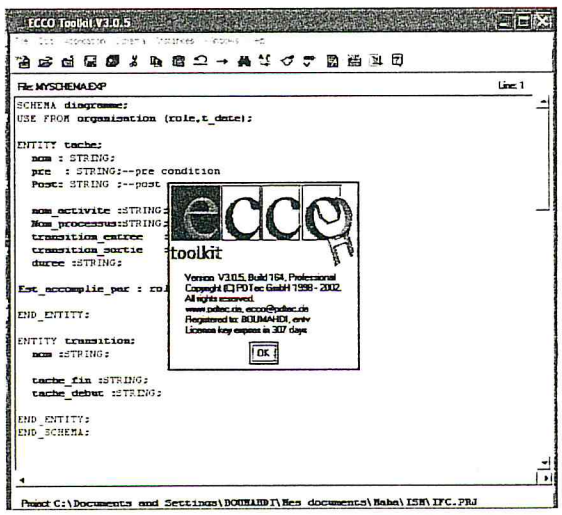


Figure C-1: L'interface de l'ECCO Toolkit

[16] RPK: Institue pour informatique appliquée dans mécanique.

B. Compilateur EXPRESS: ECCO

ECCO est un compilateur de modèle EXPRESS. C'est un logiciel sous licence. Il offre beaucoup de possibilités telles que :

- Edition, vérification de syntaxe que de sémantique des modèles.
- Manipulation graphique de la population des modèles.
- Génération d'une librairie de fonctions en java et C++ pour :
  - → L'accès aux données
  - → Lecture /écriture de fichiers physiques
  - → Vérification des contraintes
  - → D'accéder à la description du schéma lui-même sous forme d'instances d'un méta schéma d'EXPRESS
- De programmer dans le langage EXPRESS–C qui est essentiellement le langage EXPRESS et permet donc d'accéder à un modèle EXPRESS et à ses instances mais possède également deux extensions :
  - Capacité de faire des entrées-sorties (par exemple lire et écrire dans un fichier)
  - Capacité de déclencher un programme par un événement (il s'agit donc d'une sorte de « programme principal ») écrit en EXPRESS, ce qui n'existe pas en EXPRESS standard.

C. Le fonctionnement de ECCO

Les opérations d'éditions sont subordonnées à des vérifications lexicales, syntaxiques et sémantiques (statiques) des spécifications, ou modèles de données [Sardet, 99], ainsi définis. La Figure suivante illustre le fonctionnement de Ecco.

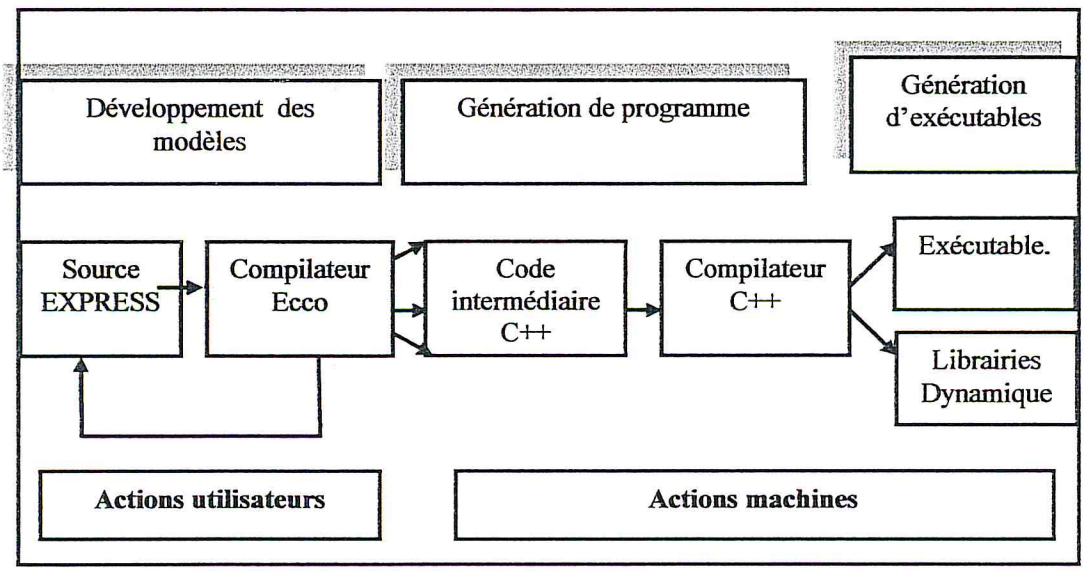


Figure C-2: Fonctionnement de Ecco [Sardet, 99]

Le code source est passé en entrée d'un analyseur en vue de la vérification de la structure du code écrit, conformément à la grammaire du langage EXPRESS, et en vue de la vérification des règles de sémantique statique, comme par exemple le contrôle de types.

Lorsque cette première phase de vérification d'un modèle de données EXPRESS est terminée, un code intermédiaire (du C++/Java) est généré, puis compilé et soumis à un éditeur de liens afin de produire un code exécutable (ou une librairie dynamique) qui va permettre de créer des populations de données.

Le fichier binaire (exécutable ou bibliothèque dynamique) généré lors de la troisième phase constitue un système de gestion de base de données [Sardet, 99], dédié au modèle de données EXPRESS dont il est issu.

D. Les composants d'ECCO Toolkit

ECCO Toolkit a une architecture modulaire pour être capable de réagir aux besoins des différents utilisateurs, et comprend plusieurs composants pour la mise en oeuvre effective de solutions d'EXPRESS (Figure C-3):

D.1 . **L'éditeur d'ecco EXPRESS:** les schémas peuvent être créés facilement et édités.

D.2 . **Le compilateur de l'Ecco** : vérifie le schéma pour convenance syntactique et sémantique et crée des applications exécutables ou bibliothèques dynamiques qui peuvent être liées aux applications externes.

D.3 . **Le contrôleur de la contrainte de l'Ecco :** est utilisé pour valider l'exactitude de chargement des données crées [Ecco, 01]. Des instances individuelles aussi bien que des populations des schémas entiers peuvent être vérifiés en utilisant plusieurs options qui sont disponibles pour l'indication du genre de vérification requis:

Les données à vérifier peuvent être choisies d'un ou d'autre plusieurs schémas et le type particulier de vérification (règles locales, règles globales, ...etc.) peut être indiqué. Les résultats de la validation du processus peuvent être sauvegardé dans des rapports.

D.4 . **API Ecco :** La communication entre les applications externes et les applications ont développé avec ECCO Toolkit est exécuté à travers ***API Ecco***. Il fournit une interface d'acces à la BDD aux langages de programmation (C++, C, TCL, Java, Visual Basic.).

D.5 . **L'adaptateur des données de l'Ecco:** fournit une interface entre les sources de données externes et les applications construites par Ecco. Il permet la communication avec les bases de données et également la lecture et l'écriture dans les fichiers physiques

D.6 . **Le moteur Ecco d'EXPRESS:** exécute les programmes établis par le compilateur d'ECCO et fournit les moyens de lire, de créer et de supprimer des données selon les schémas compilés.

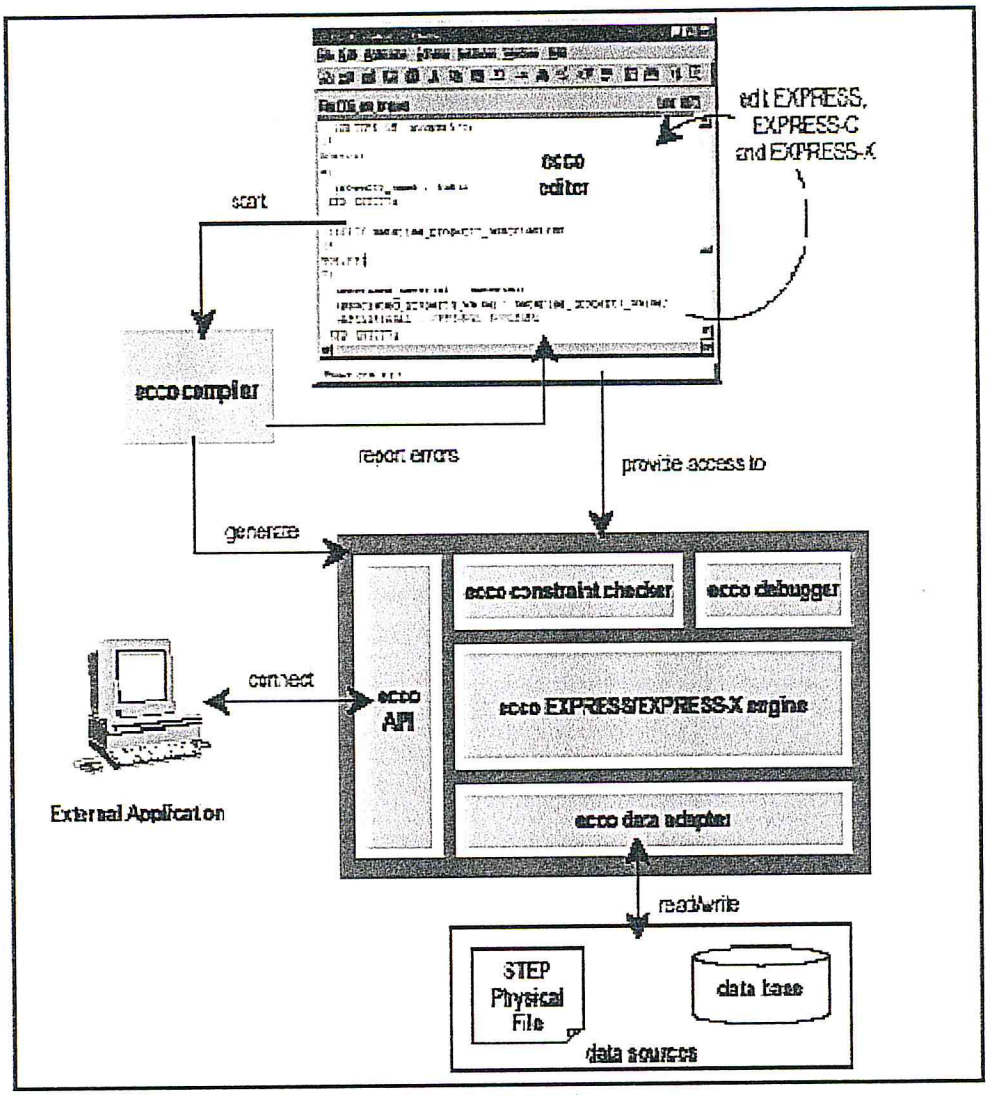


Figure C-3: L'ensemble des composants de l'ECCO toolkit [Ecco, 01]

ANNEXE D. MANUEL D'UTILISATION DE L'OUTIL DE DEFINITION

A. Définition

ODWF est un logiciel qui comprend un éditeur graphique, il permet à un utilisateur de modélser des processus, dans cette annexe nous présentons un manuel d'utilisation de cet outil. L'usage de ce logiciel demande de la part des usagers une certaine base des notions de concepts processus métier.

B. Logiciels supportés

- Visual C++ version 6.0 : il faut installer ce logiciel au moins une foie.
- ECCO Toolkit version 3.0.5: il suffit d'installer ce logiciel, et n'est pas nécessaire d'avoir une clé. Ce logiciel est disponible avec le CD d'installation de ODWF.

C. L'utilisation de l'éditeur graphique

Pour définir un processus il faut définir d'abord l'organisation et les systèmes d'information de l'entreprise, puis introduire le fonctionnement du processus.

C.1 . Définition de l'organisation

La définition de l'organisation consiste à définir les moyens humains et technologiques mis en œuvre pour réaliser les activités. Les rôles et les acteurs sont des concepts de base de l'organisation. Donc il faut déclarer tous les acteurs qui participent à l'exécution de processus modélisé, pour cela il suffit de cliquer sur l'icône approprié au définition des acteurs, une boite de dialogue s'affichera (Figure D-1).

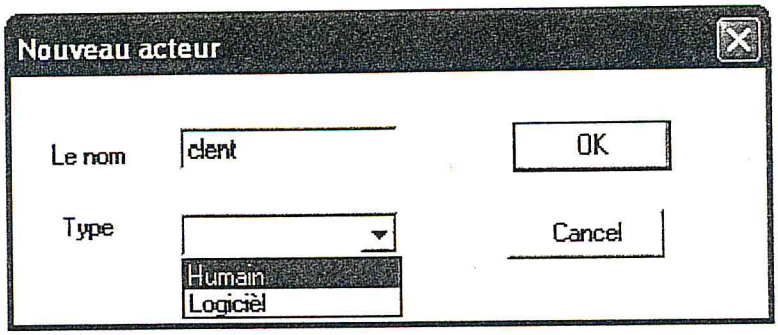


Figure D-1: Définition d'un acteur

Après la définition des acteurs, l'utilisateur de notre outil peut déclarer les rôles chargés d'exécuter les activités du processus. La figure suivante montre la déclaration d'un rôle :

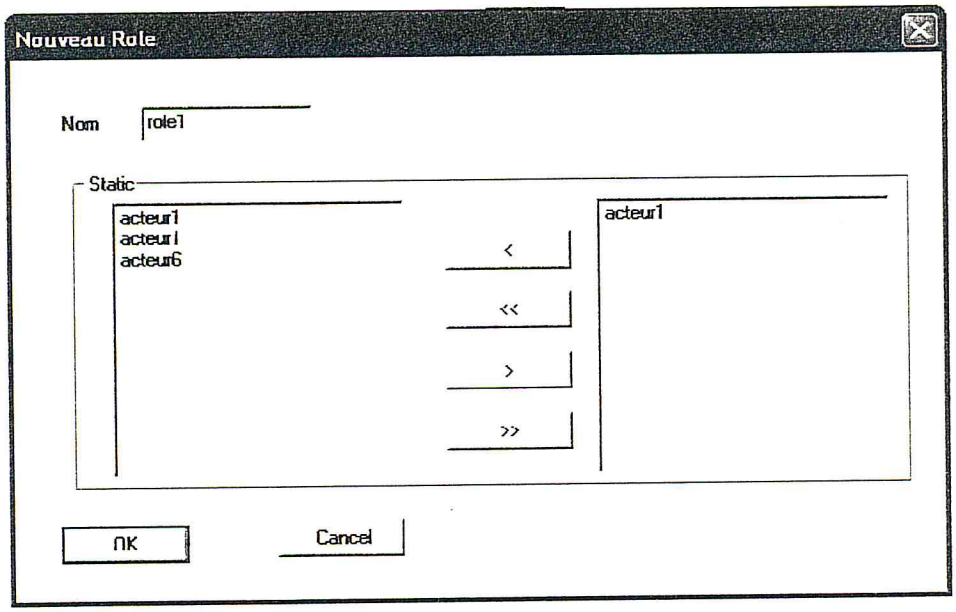


Figure D-2: Définition d'un rôle

C.2 . Définition du fonctionnement

La définition du fonctionnement consiste à définir le processus en terme de ses constituant comme: les activités, les tâches et les transitions. Et comme nous avons vu dans le méta modèle, chaque tâche est appartient à une activité, donc il est nécessaire de définir les activités avant les tâches, la figure suivante montre la déclaration d'une activité :

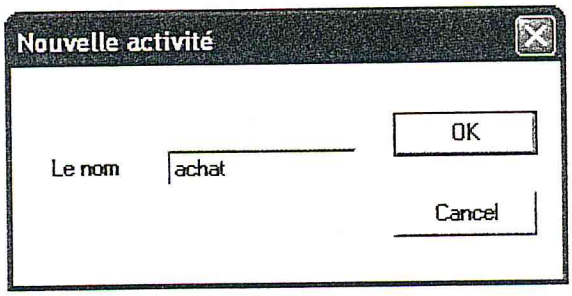


Figure D-3: Définition d'une activité

Pour la définition d'une tâche élémentaire, la tâche est représentée par un cercle, dans sa partie droite en haut on trouve le rôle qui l'exécute, et dans sa partie droite en bas on trouve la donnée qui la manipule. Chaque tâche doit être spécifié en moment de la définition de processus, un click sur l'icône d'une tâche élémentaire affiche une fenêtre des propriétés de cette tâche, pour remplir ses informations.

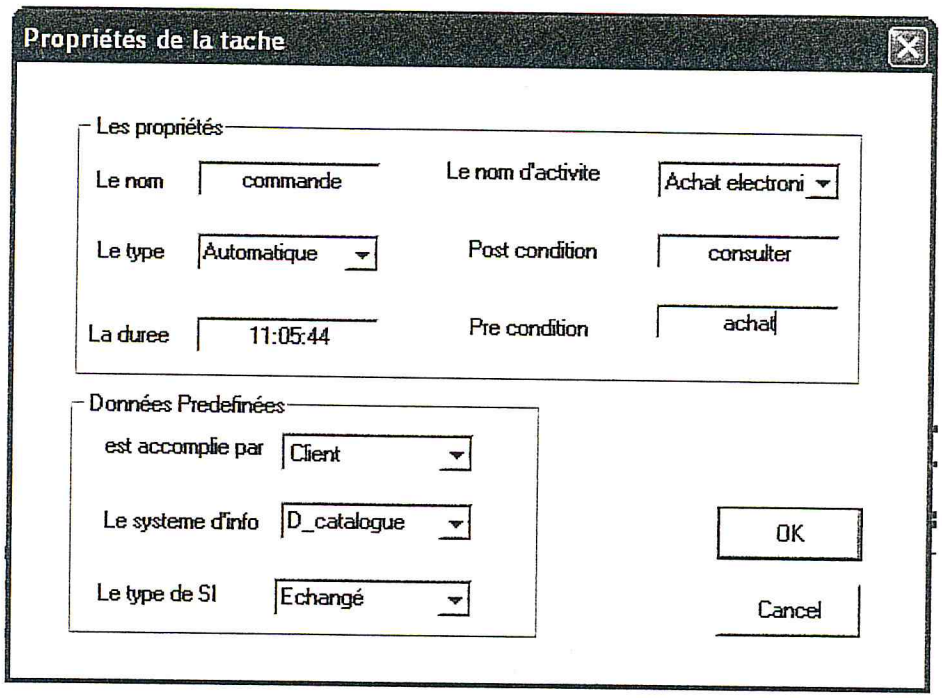


Figure D-4: Définition d'une tâche élémentaire

Après le remplissage de tous les données d'une tâche, l'utilisateur positionne le symbole de tâche (cercle) dans l'endroit ou il désir la mettre. Si l'utilisateur veut modifier ou consulter des propriétés d'une tâche, un double click sur cette tâche affiche une boite de dialogue contient tout les propriétés de la tâche.

Remarque

Les mêmes étapes sont suivies dans le cas de définition d'une tâche composée.

Un lien entre les objets s'établit par une sélection du lien, et un click sur les points de départ et d'arrivée, ceci aboutit à l'affichage d'une fenêtre qui contient des informations sur cette transition, comme montre la figure suivante :

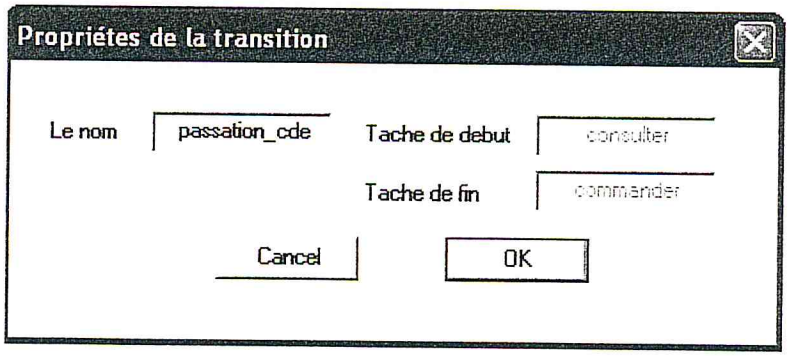


Figure D-5 : Définition d'une transition

C.3 . Définition des systèmes d'information

La définition d'un SI consiste à définir l'ensemble des entités inclut dans ce SI, au moment de la définition des SI l'utilisateur peut ajouter ou supprimer des entités comme montre les figures suivantes :

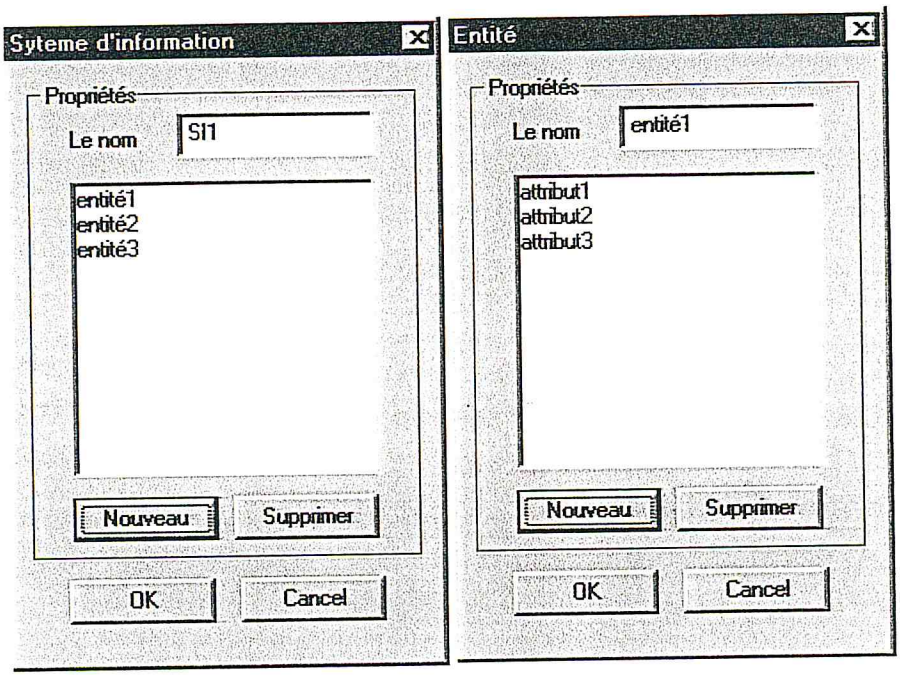


Figure D-6 : Définition des SI et des entités

Un SI est définit par son nom et ses entités, chaque entité est caractérisé par un nom et regroupe un ensemble des attributs, au cours de la définition des entités, l'utilisateur peut définir les attributs de cette entité comme il peut les supprimer. Un attribut est définit par son nom et son type (Figure D-7).

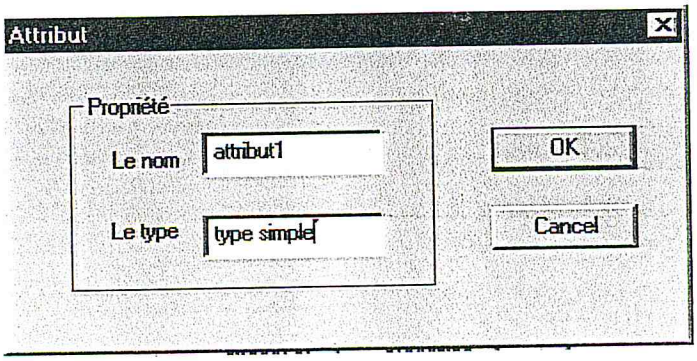


Figure D-7: Définition des attributs

D. Les fonctionnalités de l'outil de définition

➢ Pour ajouter ou supprimer un élément

• Si l'utilisateur veut ajouter un élément, il sélectionne la forme sous ou en regard de laquelle ajouter la nouvelle forme, puis il click sur l'icône de l'élément dans la barre d'outils.

• Si l'utilisateur veut supprimer un élément, il le sélectionne et il appuie sur SUPPR. Ou il choisit menu édition/supprimer (transition, tâche…) (figure ci-dessous), comme il peut utiliser un menu contextuel (click droit).

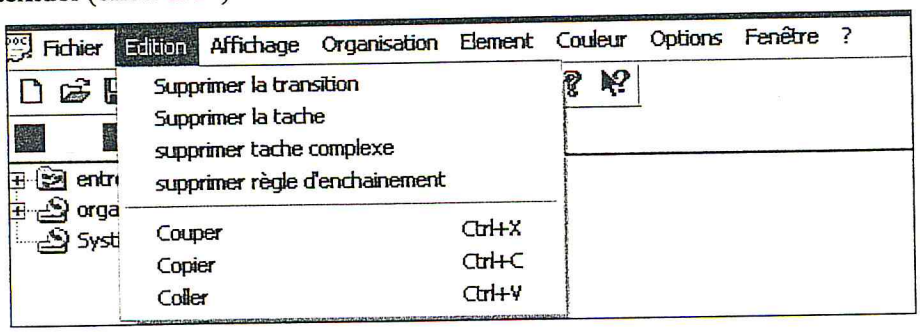


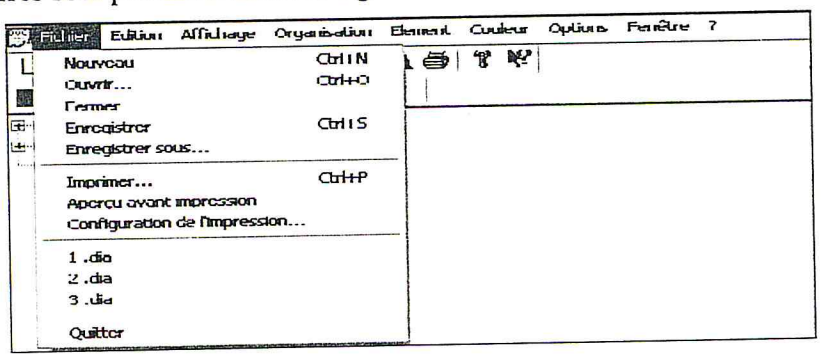
Figure D-8 : Le menu Edition

➢ Pour déplacer un élément

1. Sélectionner l'élément du modèle à déplacer.
2. Reculer l'élément.

D.1 . Les menus et des barres d'outils

Un menu affiche une liste de commandes. Des images sont placées en regard de certaines de ces commandes de façon à pouvoir associer rapidement la commande avec l'image. La plupart des menus se trouvent sur la barre de menus, qui est la barre d'outils en haut de l'écran. Les barres d'outils peuvent contenir des boutons, des menus ou une combinaison des deux. Les menus et les barres sont présentés dans les figures suivantes :



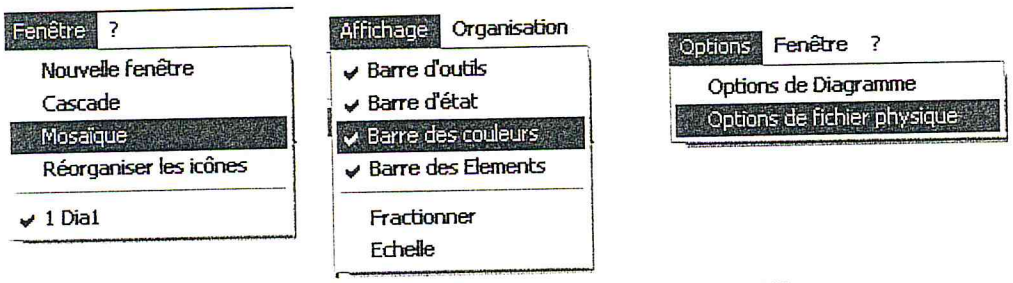


Figure D-9: Les menus de l'ODWF

D.2 . Raccourcis clavier

Nous avons intégré des fonctionnalités qui rendent notre logiciel accessible par tous les utilisateurs.

➢ Pour créer, afficher et enregistrer des modèles de processus

CTRL+N : Créer un nouveau modèle de processus.

Ou choisir fichier / nouveau.

CTRL+O : Ouvrir un document.

Ou choisir fichier / ouvrir.

Taper ou sélectionner le nom du modèle.

Choisir ok.

CTRL+W : Fermer un modèle de processus.

ALT+CTRL+S : Fractionner la fenêtre du document

ALT+CTRL+S : Supprimer le fractionnement de la fenêtre du document

CTRL+S : Enregistrer un document

➢ Pour copier et déplacer des éléments de modélisation

CTRL+C : Copier d'un élément de modélisation

Afficher le Presse-papiers

CTRL+V : Coller le contenu du Presse-papiers

CTRL+X : Supprimer l'élément sélectionné

D.3 . Les options des modèles

L'utilisateur peut personnaliser la couleur, la taille, la plume…etc. d'une tâche, d'une transition ou les règles d'enchaînement…etc. (les objets modélisés), par un click sur option de modèle dans le menu Option, la boite de dialogue qui contient ces options est représenté dans la figure suivante :

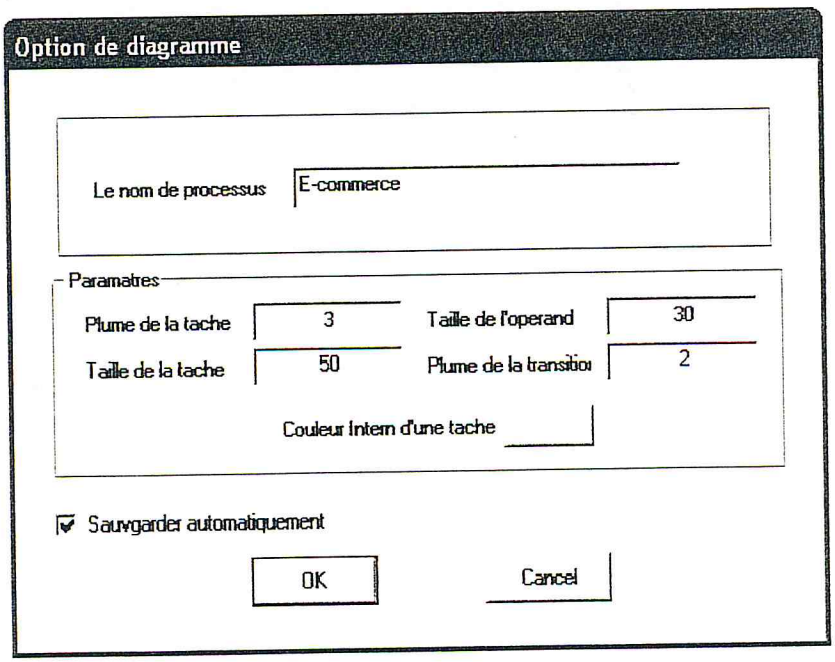


Figure D-10: Personnalisation du modèle

Les fonctionnalités offertes par notre outil ne se limite pas aux options de modèle, l'utilisateur de notre outil, peut aussi définir des options concernant la définition générée comme nous montre la figure ci-dessous :

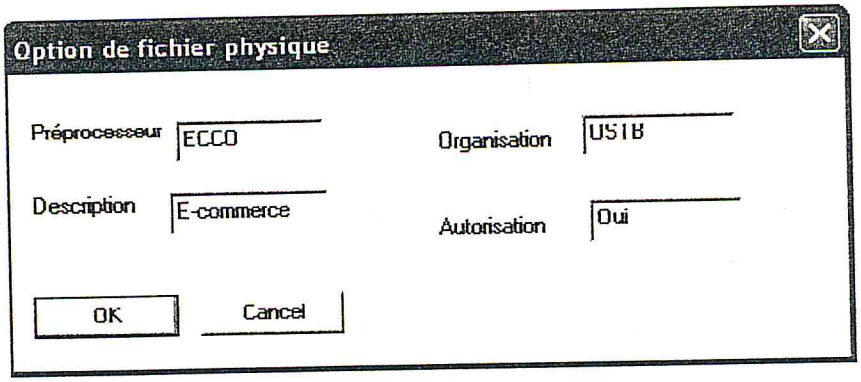
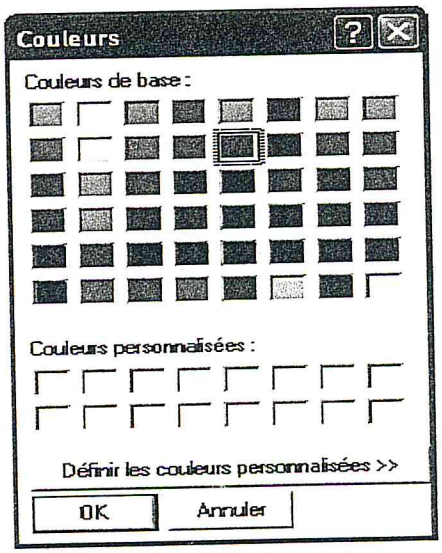


Figure D-11: Personnalisation des paramètres d'une définition

D.4 . Les options des couleurs

La palette de couleurs permettant de choisir une couleur pour les objets modélisées. La palette peut être placée n'importe où sur l'écran. Vous pouvez aussi la cacher en utilisant les commandes de menu affichage /la palette des couleurs.



D.5 . Impression et Aperçu avant impression

Pour Visualiser une page avant de l'imprimer

1. Dans le menu **Fichier**, cliquer sur **Aperçu avant impression.**
2. Utiliser les boutons de la barre d'outils pour parcourir la page ou effectuer des ajustements avant d'imprimer.

D.6 . Utilisation de l'aide

Si l'utilisateur a besoin d'aide au cours de son travail, il peut accéder à l'aide dans la fenêtre d'aide de plusieurs façons :

- Afficher le sommaire.
- Taper une question dans l'aide intuitive concernant votre modélisation d'un processus.
- Recherchez des mots ou des expressions spécifiques ou sélectionnez un mot clé à partir de la liste des mots clés dans l'index.

Analyse et conception d'un outil de définition de workflow basé sur le langage EXPRESS.

Résumé. L'arrivée des nouveaux outils de communication a entraîné de nouvelles technologies, elles comprennent le groupware, dans cette technologie, le workflow fait partie des environnements les plus puissants pour automatiser les processus de travail. Le workflow est un mode de division du travail entre les rôles, qui facilite la coordination des actions, pour cela, il faut définir un système qui gère et contrôle les différentes activités, c'est le système de gestion de workflow.

Notre travail est intéressé par la première phase de construction d'un système de gestion de workflow : c'est l'analyse de workflow. Durant cette phase il faut développer un outil qui permet la définition de workflow.

Au cours de notre projet nous avons développé un outil de définition, simple et convivial d'utilisation permettant aux usagers de représenter graphiquement leur métier, leurs activités, leurs enchaînements et les règles associées, dans un formalisme dépend du méta modèle correspondant au type d'activité modélisé.

Mots clés. Groupware, Workflow, EXPRESS, UML (unified modeling language), modélisation orienté objet , Ecco Toolkit.

Analysis and design of a workflow definition tool using the EXPRESS language.

Abstract. The arrival of the new tools of communication dragged new technologies. Among these technologies, we were interested by the groupware and especially by the workflow. The workflow makes the most powerful environment part to automate processes of work. The workflow permits the division of work between the different roles. This division facilitates the actors coordination. That is why, it is necessary to define a system of workflow management.

Our work constitutes the first construction's phase of a system of workflow management: it is the analysis of workflow. During this phase it is necessary to develop a tool that permits the definition of workflow.

During our project we developed a tool of definition, simple and convivial, allowing users to represent their profession, their activities, their sequences and the associated rules, in a formalism closed by the meta models corresponding to the type of a modal activity.

Key words. Groupware, workflow, EXPRESS, UML (unified modeling language), modeling oriented object, Ecco Toolkit.