CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE BASSE-NORMANDIE

_____

MEMOIRE

Présenté en vue d'obtenir

Le DIPLOME D'INGENIEUR C.N.A.M.

en

INFORMATIQUE

par

Jean-Yves PUCELLE

_____

Représentation Interactive de Processus Métier.

_____

Les travaux relatifs au présent mémoire ont été effectués au GREYC (Groupement de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (UMR 6072) sous la direction de Monsieur Bernard MORAND).

TABLE DES MATIERES.

REMERCIEMENTS.

Je remercie Messieurs Régis Carin (directeur) et Etienne Grandjean (directeur adjoint) de m'avoir accueilli au sein du GREYC.

Je tiens également à remercier les membres de l'équipe ISLand , sa responsable Mme Anne Nicole et particulièrement M. Bernard Morand qui est à l'initiative de ma venue pour ce stage.

Mes plus vifs remerciements vont à mes proches qui m'ont soutenu durant toutes ces années studieuses.

INTRODUCTION.

La gestion des processus métier de l'entreprise (en anglais BPM : Business Process Management) est, ces dernières années, un des grands sujets de préoccupation des entreprises. Elles y sont poussées par leur constante nécessitée d'optimiser leurs processus pour rester compétitives sur leurs marchés mais également pour des raisons d'ordre réglementaire comme la certification qualité par la norme ISO 9001 ou la réforme Bâle II sur les risques opérationnels dans le secteur financier.

La base de toute démarche de BPM consiste en la description des processus métiers de l'entreprise. C'est à dire de : "*l'ensemble des activités internes d'un métier dont l'objectif est de fournir un résultat observable et mesurable*" [Ket98]. D'abord sous forme textuelle, les processus sont de plus en plus modélisé par des représentations diagrammatiques avec l'avènement de nombreux standards et méthodes dédiées.

De plus en plus d'éditeurs de logiciels de BPM (Axway, Boc, Computer Associates, IDS Scheer, IBM, Ilog, Intalio, Mega, Popkin, SeeBeyond, ... pour ne citer que quelques-uns uns des principaux) offrent des produits qui permettent la modélisation mais aussi l'exécution ou la simulation.

C'est ce dernier point qui est le thème du projet de recherche entre France Télécom et le GREYC (Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de l'Université de Caen) intitulé : "Représentation interactive de processus métier" dont ce mémoire est le fruit du travail d'un stage de huit mois dans ce cadre.

Le constat de départ stipule que les représentations graphiques sont trop abstraites pour se faire une idée précise du comportement des processus en cours d'exécution. Elles masquent à l'observateur la vision des différentes instances de processus qui s'y déroulent. Elles ne permettent pas de se rendre compte des répercussions qu'une ou plusieurs modifications peuvent avoir sur l'ensemble qui a été modélisé. La simulation devrait répondre à ces préoccupations. Mais force est de constater que les offres actuelles sont très décevantes. Il en existe actuellement deux grandes familles:

- dans la première, les simulations consistent à évaluer les processus modélisés par un simulateur de type 'boîte noire' qui délivre des quantités de données récupérables dans des feuilles de calcul. Il faut définir auparavant le scénario qu'on souhaite évaluer, qu'il faudra faire rejouer autant de fois qu'on aura de modifications à apporter aux processus. Mais, surtout, on ne voit rien s'animer sur l'écran.

- dans la seconde famille, les simulations sont de type visuel. Mais, la plupart des offres ne montre qu'une animation des éléments du modèle par des changements de couleurs sur des exécutions successives. Il n'y a pas de possibilité de soumettre un scénario, ni de modifier les processus en cours de simulation.

Le projet se base sur cette constatation pour proposer d'aller plus en avant sur ce plan en appliquant les techniques actuelles du domaine de l'animation visuelle et sonore à la simulation de processus métier. Il souhaite y ajouter la possibilité d'effectuer des modifications directement depuis l'animation et, ainsi, de se rendre compte visuellement et dynamiquement des conséquences sur les acteurs et les instances de processus. Ce type de simulation semble parfaitement adapté à l'évaluation des évolutions des processus métier au sein d'une entreprise et sera tester sur le cas d'application intitulé "Représentation de processus métier d'une Agence Résidentiels France Télécom.". Il peut également s'adapter à une démarche commerciale de vente de biens ou de services professionnels, c'est ce qui sera évalué dans le second cas d'application de "Personnalisation de l'offre de service télécoms auprès de PME".

Le but est donc de créer un système informatique de représentation interactive de processus métier. Il sera basé sur un Système Multi-Agents (SMA) qui animera les constituants du processus. Les modifications en interaction avec la représentation imposeront une architecture sous-jacente suffisamment flexible pour permettre toutes les reconfigurations nécessaires en cours de simulation. Il sera créé des questionnaires intelligents pour déterminer rapidement quels sont les desiderata de l'utilisateur en matière de changements par rapport à l'animation qu'on lui présente à l'écran. Pour que cette dernière soit la plus pertinente possible,

une étude déterminera une panoplie d'objets graphiques en parfaite adéquation avec les éléments de processus métier que l'on souhaite représenter.

Le travail relaté dans ce mémoire s'est porté sur les points suivants :

- L'étude des différentes notations disponibles actuellement en vue de déterminer celle qui sera adoptée pour les descriptions des processus métier dans le système cible. Nous aurons à l'esprit qu'elle sera destinée à des experts métiers non-informaticiens. Il faudra qu'elle puisse s'adapter aux processus externes, qu'elle tienne compte des ressources, qu'elle permette de spécifier des scénarios,... Une part importante de l'étude sera consacrée à l'enrichissement de cette notation en vue de créer des modèles exécutables. On sait qu'aujourd'hui aucun modèle ne propose ces fonctionnalités qui sont nécessaires à toute animation de processus.

- La recherche d'une architecture assez flexible pour répondre aux besoins de modifications dynamiques de la représentation et de la description des processus en cours de simulation. Elle devra être compatible avec l'utilisation d'un Système Multi-Agents comme moteur de la simulation.

- Les spécifications fonctionnelles d'une solution de plate-forme de Représentation Interactive des Processus Métiers, viendront clôturer ce mémoire. Elles donneront notre vision de la mise en application des différents points vus dans les deux études précédentes. Elle servira de base à l'élaboration de la première version du prototype logiciel.

Pour une parfaite compréhension du travail effectué, nous invitons le lecteur à prendre connaissance dés à présent du document préparatoire au projet qui se trouve dans l'annexe I.

1. LES NOTATIONS DIAGRAMMATIQUES.

La base de notre projet sur la représentation interactive de processus métier consiste en l'étude des différents standards de notations existants. Les objectifs sont de déterminer :

- s'il existe une notation plus adaptée à notre projet que les autres.
- si elle couvre l'ensemble des besoins et, sinon, comment y remédier.

Nous allons donc tout d'abord nous intéresser aux deux principaux standards que sont UML (Unified Modeling Language) et BPMN (Business Process Modeling Notation). Puis nous aborderons des aspects complémentaires que nous fournissent deux techniques de modélisation de processus : OSSAD et ADONIS.

1.1. U.M.L. (Unified Modeling Language).

C'est un langage de modélisation orientée objet qui est issu du regroupement de trois techniques de modélisation, la méthode Booch, OMT (Object Modeling Technique) de James Rumbaugh et OOSE (Object-Oriented Software Engineering) d'Ivar Jacobson. C'est depuis 1997 un standard de l'OMG (Object Management Group (http://www.omg.org/ et plus précisément http://www.uml.org/)).

U.M.L couvrant l'intégralité du domaine du développement objet, nous ne nous intéresserons qu'aux vues qui permettent la représentation de processus métier. Il s'agit des :

- Diagrammes de cas d'utilisation (USE CASES).
- Diagrammes de Séquence.
- Diagrammes de Collaboration.
- Diagrammes d'activité.

1.1.1. Le diagramme de cas d'utilisation (USE CASES).

C'est la seule vue statique dont nous ayons besoin. Elle permet d'avoir une vue externe de haut niveau du système en interaction avec son environnement.

On trouve les acteurs (humains ou non) qui y jouent un rôle précis. Un cas d'utilisation spécifie de façon très abstraite (en une phrase) ce que le système fait suite à une sollicitation précise d'un acteur. Il peut y avoir un seul ou plusieurs cas d'utilisation regroupés en un paquetage. Cet ensemble décrit les objectifs du système.

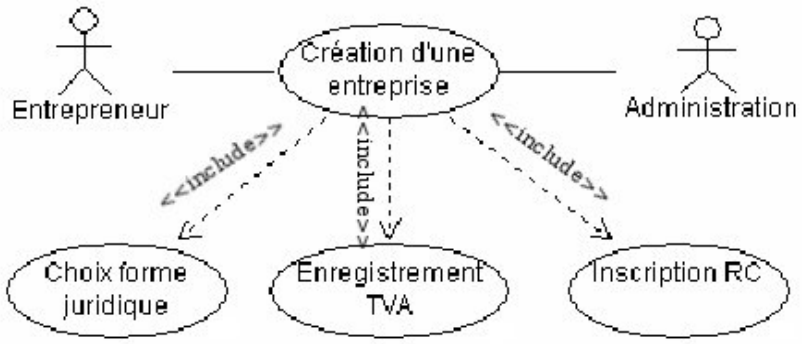


Figure I.1 Diagramme des cas d'utilisation.

Ce diagramme permet une certaine décomposition grâce au système d'inclusions et d'extensions noté par une flèche en pointillés.

Du point de vue de la représentation des processus métier, il permet de modéliser les processus dits abstraits avec leurs acteurs.

1.1.2. Le diagramme de séquence.

Il forme, avec le diagramme de collaboration, les diagrammes d'interaction de UML. Le diagramme de séquence montre l'organisation temporelle des échanges entre les objets du système. Ces objets sont matérialisés par leurs lignes de vies en pointillés qui devient un trait épais lorsque l'objet est actif. Le temps s'écoule suivant l'axe vertical de haut en bas.

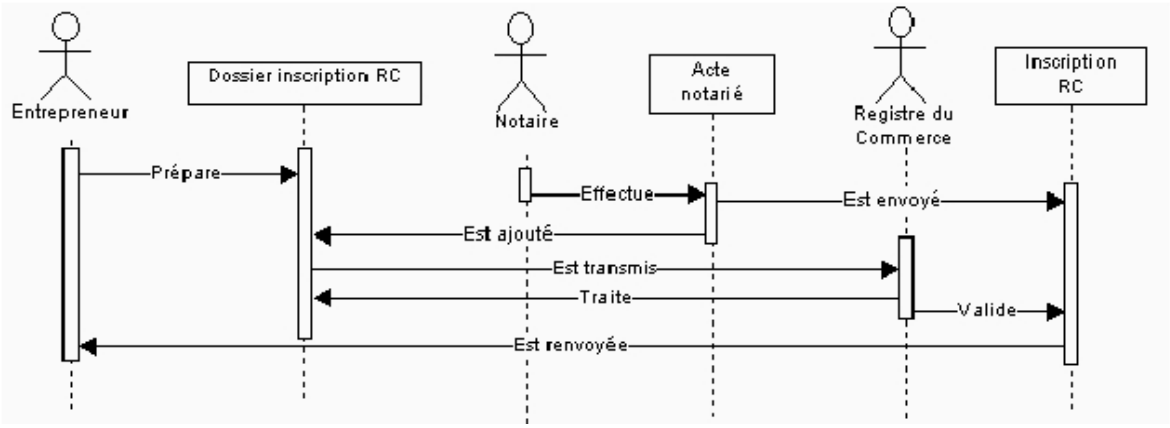


Figure I.2 Diagramme de séquence.

1.1.3. Le diagramme de collaboration.

Le diagramme de collaboration est sémantiquement équivalent au diagramme de séquence.

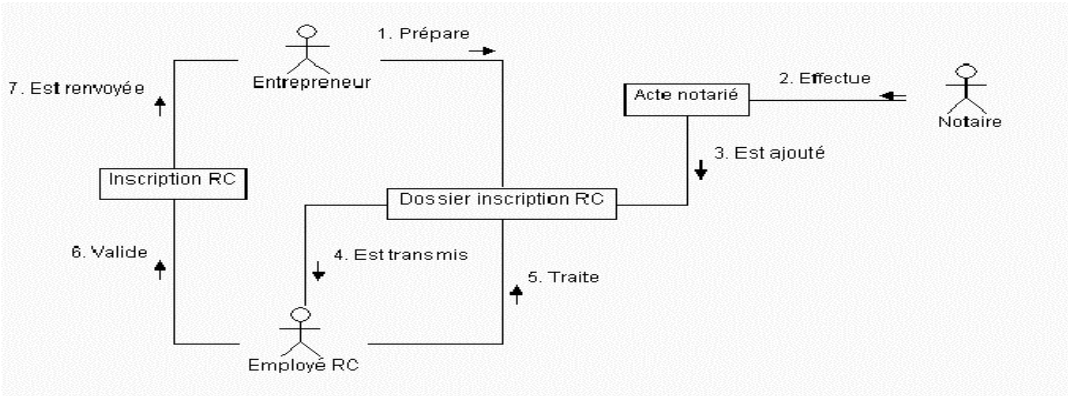


Figure I.3 Diagramme de collaboration.

Il montre également les échanges entre les objets du système mais en s'attachant à montrer l'organisation spatiale.

Un aspect d'ordonnancement est donné par une numérotation des messages.

1.1.4. Le diagramme d'activités.

Il décrit le déroulement d'un processus formalisé le plus souvent dans un cas d'utilisation. Les activités sont représentées par des 'boîtes' reliées par des flèches de transition. On marque un début et une ou plusieurs fins.

Au niveau des structures de contrôle, on trouve la décision symbolisée par un losange. Il y a aussi la barre horizontale, qui suivant si les flèches divergent ou convergent, matérialise le parallélisme ou la synchronisation.

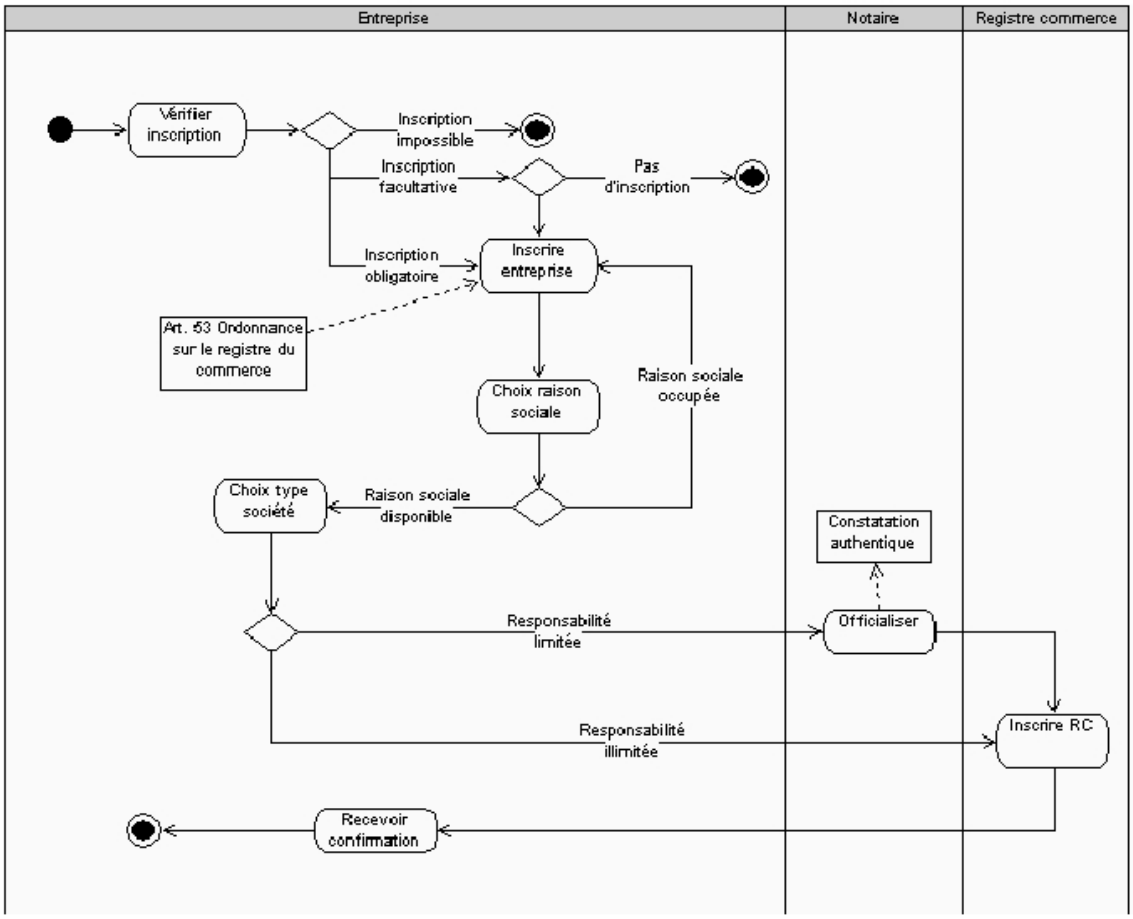


Figure I.4 Diagramme d'activités.

Il est possible d'y montrer des flots d'objets (tels que des factures, des formulaires,...) avec leurs états successifs.

Les couloirs d'activités apportent une dimension organisationnelle en attribuant la responsabilité des actions à accomplir à l'entité spécifiée en tête du couloir.

1.1.5. Conclusion sur UML.

Nous reprochons, tout d'abord, à UML le découpage en plusieurs modèles qui, selon nous, n'est pas adaptable à l'utilisation par un expert métier telle que nous la souhaitons dans le projet. Dans notre système, nous souhaiterions que l'expert décrive les processus par le biais d'une interface unique donc, à priori, sur un seul modèle. Ceci pour faciliter les modifications mais aussi pour lui éviter les "va et vient" entre plusieurs modèles.

Elle est aussi très générique et marquée par sa vocation de langage de modélisation objet dont l'objectif est l'implémentation informatique.

Le manque de spécialisation de certains éléments de la notation comme les événements ou les activités laisse à penser que certains processus métiers ne pourront pas être modéliser convenablement par ce langage. La sémantique trop faible, notamment au niveau du diagramme des activités, oblige à masquer certains aspects spécifiques des processus métiers. C'est le cas, entre autres, des annulations, des règles métiers, de certaines décisions complexes.

La possibilité d'utiliser des stéréotypes, qui sont des extensions du méta-modèle par l'utilisateur, permettrait d'adapter UML à la modélisation des processus métier mais ce serait au détriment du standard.

1.2. B.P.M.N. (Business Process Modeling Notation).

Il s'agit d'une notation spécialisée processus métier. C'est un standard relativement 'jeune' (version 1.0 de mai 2004), qui ne s'intéresse pas directement à l'implémentation. Le but recherché est de proposer une notation qui permet aux experts métiers de décrire de façon graphique leurs processus. Elle vient donc remplacer toutes les descriptions textuelles non standardisées qui étaient utilisées jusqu'alors. La raison qui sous-tend est de faire collaborer les décideurs métiers, qui ont la maîtrise d'ouvrage de leurs processus et les informaticiens qui en sont les maîtres d'œuvre. Elle est développée par le BPMI (Business Process Management Initiative (http://www.bpmi.org/)).

Dans BPMN il n'y a qu'un seul diagramme appelé un BPD (Business Process Diagram). Il a de nombreuses similitudes avec un diagramme d'activités d'UML. C'est d'ailleurs une des volontés de départ des membres du BPMI et certains pensent que les deux standards évolueront conjointement, le BPMI étant membre de l'OMG.

L'approche est faite sur un seul plan celui des processus. On part de ceux de plus hauts niveaux et on établit une sorte de hiérarchie de sous-processus pour atteindre le degré de granularité le plus fin que sont les tâches. C'est une approche dite 'top-down'.

Au niveau d'abstraction le plus haut on ne trouvera que des processus abstraits ce qui donne pratiquement la même vision que dans un diagramme des cas d'utilisation d'UML.

De plus, les messages échangés y figurant également, le BPD reprend sur le même diagramme une grande partie de la sémantique des diagrammes d'interaction d'UML.

Mais BPMN apporte de nombreuses autres améliorations à la notation graphique de processus métiers.

1.2.1. Les activités.

Il y a tout d'abord la tâche élémentaire, c'est à dire non décomposable et que l'on ne peut pas interrompre en cours. Elle est représentée par un rectangle aux bords arrondis avec, au centre, un texte court donnant une description de ce qui s'y passe.

Du fait du principe de composition-décomposition, on trouve le sous-processus qui, quand il est dans sa forme réduite, se note comme une tâche mais en y ajoutant le symbole '+'.

D'autres spécificités peuvent être adjointes aux activités :

- la répétition de type boucle qui correspond à la forme algorithmique 'tant que'.

- la répétition de type multi-instances correspondante à un 'pour chaque'.

- la compensation marque une activité associée à une autre et déclenchée dans le cas d'une annulation ou de l'échec d'une transaction. La compensation rétablie la situation qui existait avant le début de l'activité à laquelle elle est associée.

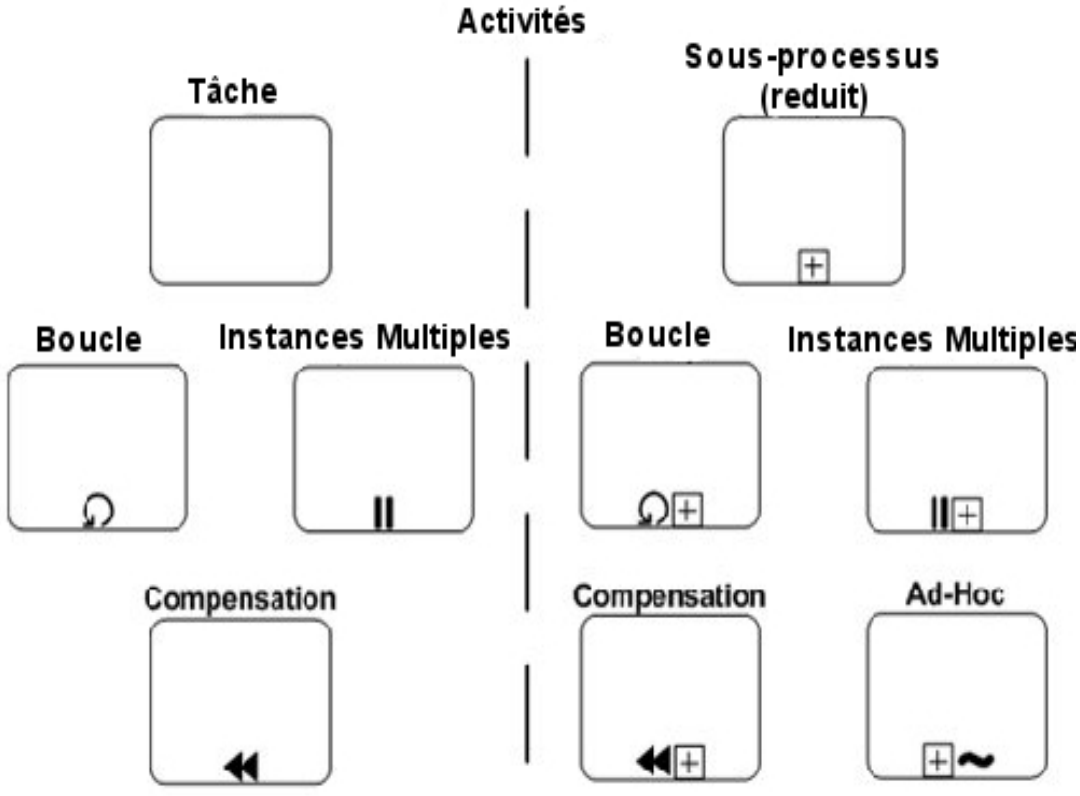


Figure I.5 Les types d'activités.

Le type ad-hoc correspond à un groupe d'activités dont l'ordonnancement précis ne peut être déterminé à l'avance. C'est donc un type de sous-processus uniquement. Pour

l'illustrer, on peut donner l'exemple de la rédaction d'un livre. L'auteur doit taper le texte, le mettre en forme, chercher ou créer des figures. C'est un processus ad-hoc car il choisit selon sa convenance l'ordre de ces tâches qui peut varier d'un livre à un autre mais également en fonction de l'auteur.

Dans sa forme développée un sous-processus est délimité par un rectangle pouvant porté le pictogramme de son type.

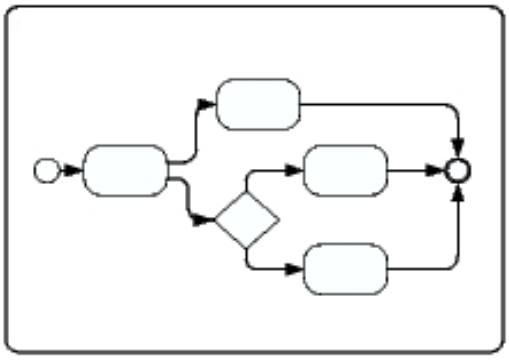


Figure I.6 Forme développée du sous-processus.

1.2.2. Les événements.

La notation BPMN est riche au niveau de la spécification des événements. Elle les classe en trois catégories :

- Début : ce sont les événements reçus qui déclenchent un processus.

- Intermédiaire : ils interviennent au cours du processus sans pour cela le déclencher ou le terminer. Ce sont des événements qui affectent le cours du processus lors de leurs survenues. Ils peuvent être reçus ou émis.

- Fin : tous les événements qui terminent un processus ou un sous-processus. Ils sont soit une fin totale du processus soit la fin d'un sous-processus avec l'émission d'un événement vers le sous-processus suivant.

Ces trois catégories sont déclinées en neuf types:

- Message : correspond aux messages échangés entre les participants.

- Temps : permet de noter des événements de type temporel (ex: après 2 heures; tous

les lundis à 9H,...).

- Exception : sert à la prise en charge d'erreurs.

- Annulation : permet de spécifier qu'une possibilité d'annulation est envisagée. On la trouve le plus souvent dans des sous-processus ayant des transactions.

- Compensation : sert à associer un sous-processus qui supprimera les modifications effectuées suite à une annulation ou à l'échec d'une transaction.

- Règle : matérialise le déclenchement d'une règle métier. (ex: progression du CAC40 > 10%; stock < 10 unités).

- Lien : permet de relier directement la fin d'un processus au début d'un autre généralement dans un autre BPD.

- Terminaison : indique que l'on a atteint la fin complète du processus.

- Multiple : est utilisé pour indiquer que plusieurs événements de types différents peuvent intervenir en produisant les mêmes effets.

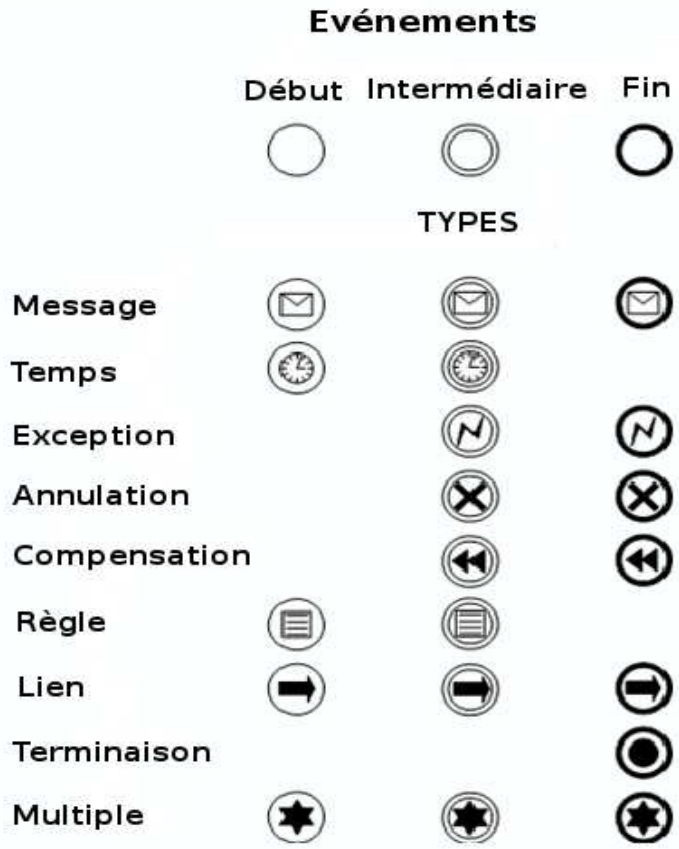
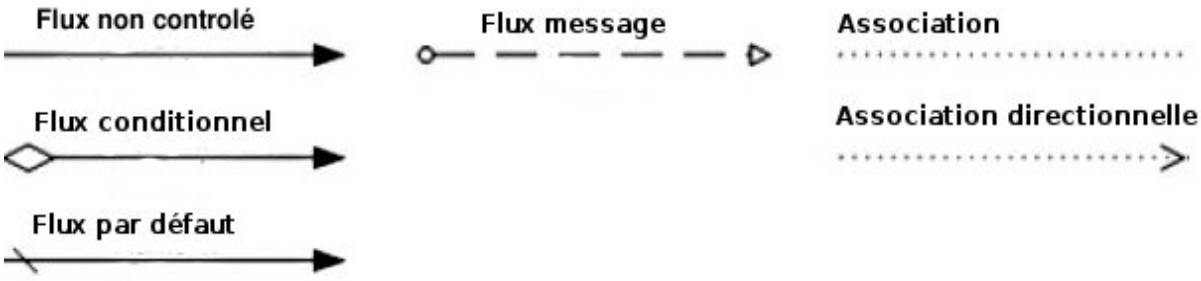


Figure I.7 Diagrammes d'événements.

On remarque que, bien évidemment, tous les types ne peuvent pas être déclinés dans chaque catégorie.

1.2.3. Les connecteurs.

Ils servent à montrer l'organisation des différents flux dans un processus. On compte trois catégories de connecteurs dans BPMN, les flux de séquences, les flux messages et les



associations.

Figure I.8 Les connecteurs.

Les flux de séquences matérialisent l'enchaînement des activités pour réaliser le processus. Ils existent sous trois formes. Le flux non contrôlé qui fait un enchaînement direct. Le flux conditionnel qui permet de définir une expression (on parle alors d'une garde) qui doit être vérifier pour pouvoir emprunter cette voie. Cette forme permet également de gérer les cas où les conditions de sorties d'une activité sont multiples et non-exclusives (équivalent à la structure algorithmique du 'case of'). Le flux par défaut vient la compléter en indiquant quel chemin prendre dans ce cas où aucune des gardes des flux conditionnels n'est vraie. A notre sens cela fait double emploi avec la structure de contrôle du type 'OU' inclusif disponible que nous verrons plus loin.

Les flux messages sont utilisés pour montrer les échanges de messages entre deux entités organisationnelles matérialisées par des couloirs d'activités. Ces flux s'arrêteront à la frontière du couloir si l'on ne détaille pas les activités qui s'y déroulent. Dans le cas contraire, ils relieront les activités impliquées chez l'une et l'autre.

Les associations servent à rajouter des informations au BPD comme, par exemple, des annotations. L'association directionnelle sera utilisée conjointement à des objets de données (formulaires, factures,...) pour en montrer le parcours.

1.2.4. Les structures de contrôle.

Comme pour les évènements, les quatre natures de structures de contrôle se différencient par l'adjonction d'un pictogramme au centre de la figure de base qu'est le losange. Une courte description donne la décision qui y est prise.

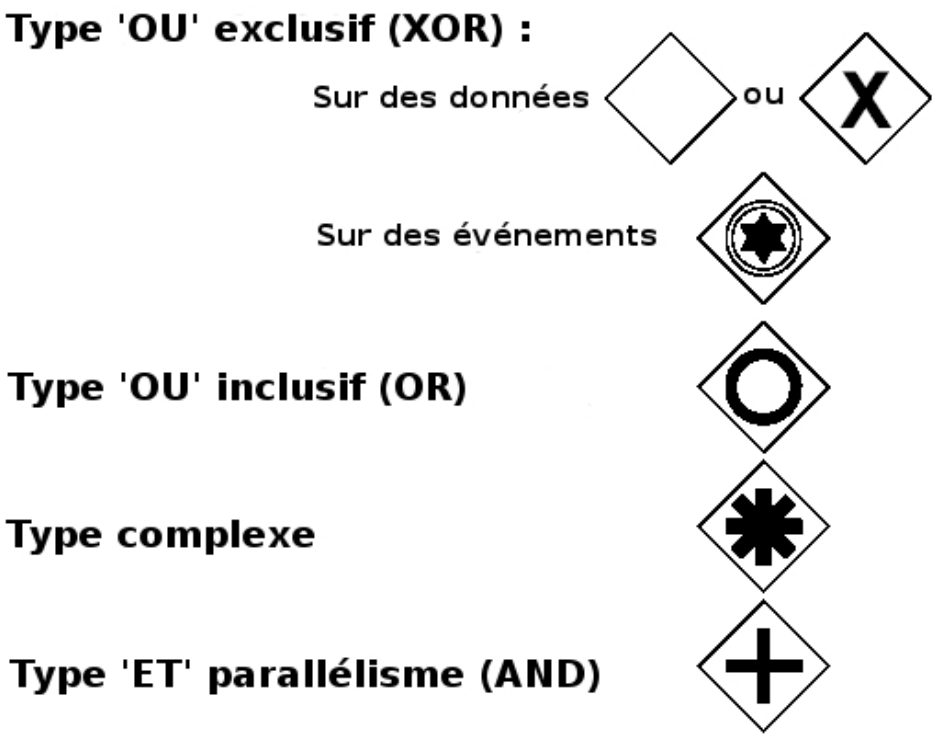


Figure I.9 Les structures de contrôle.

Le type OU exclusif (XOR) est une structure qui permet de montrer les différents chemins que le flux peut prendre à un point précis de décision. Mais, seulement un sera choisi. Il est de deux natures, soit il porte sur des valeurs de données du processus comme par exemple le montant d'une commande pour une remise client. Soit, il s'agit d'évaluer la nature d'événements reçus : une réponse par 'oui' ou par 'non' d'un organisme de crédit.

Le type OU inclusif (OR) est celle qui permet d'évaluer tous les chemins possibles de façon non exclusive. On peut se représenter cette structure comme le regroupement de plusieurs alternatives élémentaires indépendantes. Ainsi, tout ou partie des chemins seront empruntés. Il faut toujours prévoir un chemin par défaut dans le cas où aucune des conditions ne serait vraie.

Le type complexe sert à définir des décisions qui sont des compositions des différents types précédents. Il peut aussi être utilisé dans le cas d'expressions complexes que l'on ne peut réaliser avec les types de base, comme par exemple, un minimum de conditions vraies parmi une liste.

Le type ET : le parallélisme (AND) crée et synchronise des séquences en parallèle. Pour la synchronisation, il peut s'agir de séquences créées par une structure de type ET ou de type OU inclusif.

1.2.5. Les objets symboliques (Artefacts).

Ce sont des notations supplémentaires qui permettent de rajouter des informations complémentaires servant à améliorer la compréhension du processus modélisé.

Les objets de données sont particulièrement intéressants puis qu'ils donnent la possibilité de matérialiser des objets qui sont utilisés et modifiés par le processus. Ce sont des documents (électroniques ou non), des données ou d'autres objets qui peuvent être associés à un flux de séquence. Mais aussi, en entrée ou en sortie d'une activité à l'aide d'une association directionnelle.

Le standard a prévu la possibilité de notifier leurs états successifs. Cette fonction est très importante puisque certains experts métier, principalement du tertiaire, ont une vision de leurs processus qui est du type "document-état". Les objets de données leur permettent que spécifier les états successifs des documents tout en restant dans une approche "événement-action" qui est la base de BPMN.

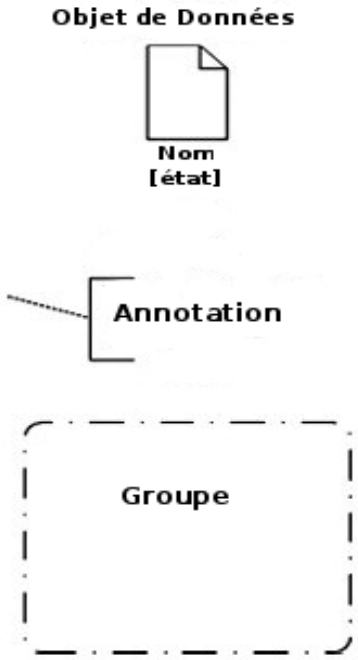


Figure I.10 Les objets symboliques.

L'annotation permet au rédacteur d'ajouter toutes les informations qu'il souhaite et de les relier à la partie du processus concernée.

Enfin les groupes servent à délimiter une partie du processus que l'on souhaite nommer ou annoter.

1.2.6. Les couloirs d'activités.

On retrouve ici le même principe que dans les diagrammes d'activités d'UML qui consiste à utiliser des couloirs pour montrer les responsabilités de chaque acteur dans le processus.

Ils se présentent sous deux formes selon le degré de décomposition souhaité. Il y a le type 'Pool' où le couloir représente un participant au processus. Ce peut être une entité organisationnelle comme une entreprise ou un rôle générique (client, fournisseur,...).

Le type 'Lanes' permet d'affiner la vision précédente en y ajoutant des divisions. Le standard n'impose pas d'usage particulier de ces bandes. Elles sont généralement utilisées pour représenter des rôles internes (directeur, secrétaire,...) ou des départements au sein de la même

organisation.

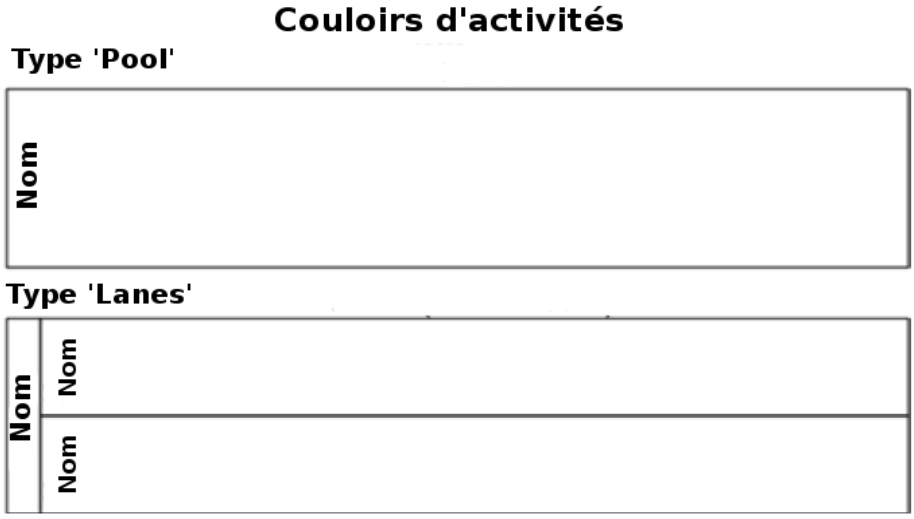


Figure I.11 Les couloirs d'activités.

1.2.7. Exemple de BPD.

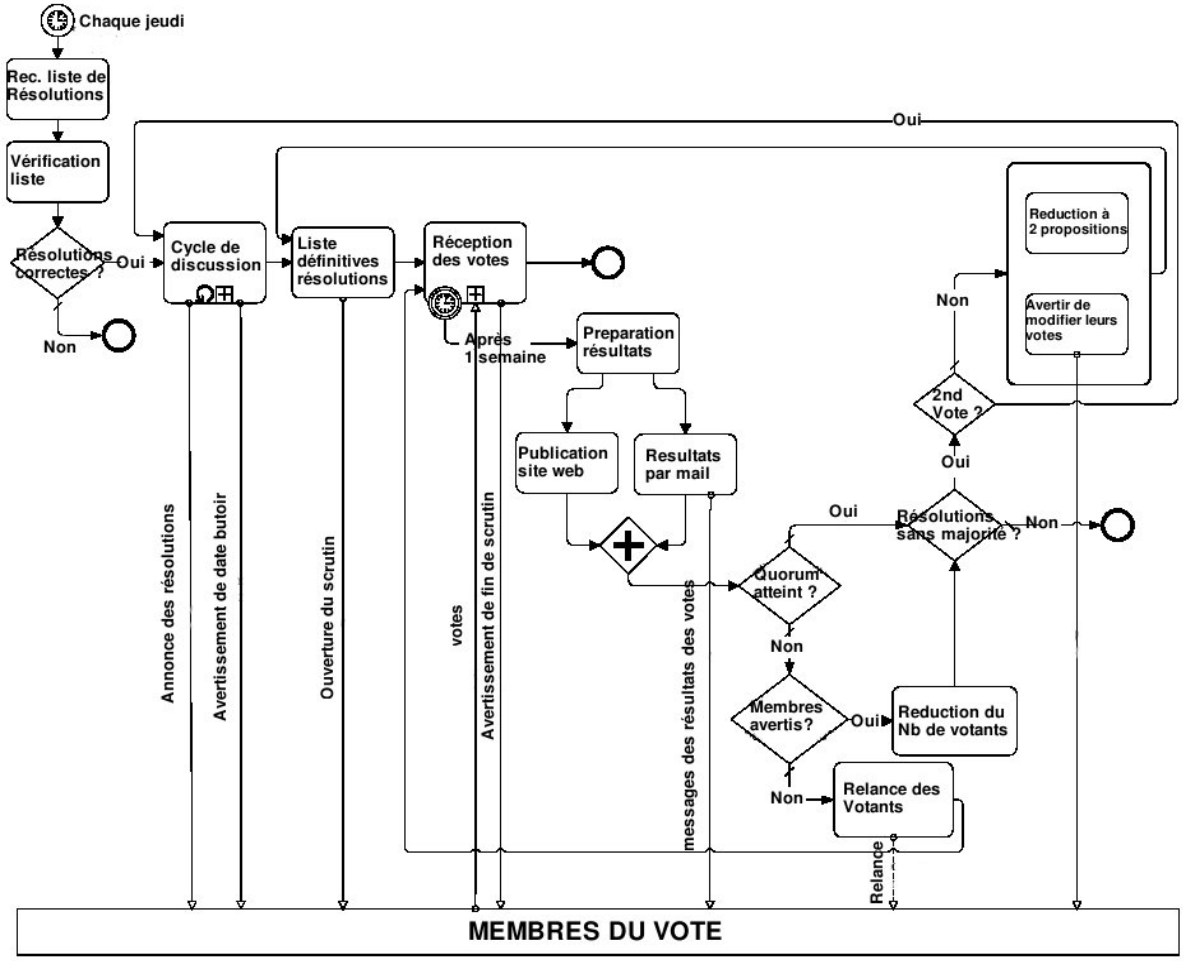


Figure I.12 BPD d'un processus de vote électronique.

1.2.8. Conclusion sur BPMN.

BPMN offre une notation qui couvre plus correctement les besoins en modélisation de processus métier qu'UML. Les avancées au niveau de la spécialisation des événements, des structures de contrôle et de la prise en considérations de phénomènes comme les annulations et les compensations font de BPMN la notation la plus spécialisée dans ce domaine. C'est d'autant plus remarquable que la gamme des diagrammes est peu étendue. Elle permet, de la sorte, aux experts métier de l'acquérir rapidement. En s'inspirant largement des diagrammes d'activités d'UML, elle sera facilement compréhensible par des informaticiens déjà utilisateurs du langage de modélisation de l'OMG.

Son plus grand avantage se situe au niveau de la spécialisation des activités et des événements. BPMN répond aux besoins de modélisation des processus métiers actuels basés sur le workflow ou contenant des transactions comme c'est le cas pour le commerce électronique (Business to Business, Business to Consumer,...)

La vision des processus sur plusieurs niveaux est appréciable pourvu qu'on utilise une version électronique du BPD. Cela permet d'aller du général (processus abstraits) au particulier en gardant une sémantique commune.

Nous regrettons que le mapping avec le langage d'exécution BPEL (Business Process Execution Language) qui est défini dans la norme n'a pas encore vraiment pris forme dans les produits commerciaux. Et plus généralement, qu'il ne soit pas défini de format d'exportation similaire à XMI (XML Metadata Interchange) qui est la syntaxe XML d'échange pour les modèles UML. Il faut incontestablement que la communauté BPMN fasse des efforts dans ce sens pour imposer sa notation.

Pour nos besoins en simulation et animation, si elle s'avère satisfaisante pour la description opérationnelle des processus métier, le pouvoir d'expression de la partie organisationnelle, qui n'est guère plus fort que dans UML, ne nous convient pas. BPMN ne permet qu'une division des entités organisationnelles en rôles. Il lui manque la possibilité de modéliser les acteurs réels et les ressources organisationnelles.

1.3. Les apports d'OSSAD et d'ADONIS.

Il est intéressant de se pencher sur ces deux techniques de modélisation de processus parce qu'elles ont développé considérablement la vision organisationnelle plutôt réduite dans les deux notations précédentes. Ceci est particulièrement bien montré par Glassey et Chappelet dans leur étude [GC02].

C'est un point que nous devons développer pour notre projet pour, par exemple, prendre en compte des modifications liées à la structure organisationnelle.

1.3.1. OSSAD (Office Support System Analysis and Design).

C'est une méthode élaborée fin des années 1980 lors du programme européen ESPRIT (European Strategic Program for Research in Information Technology).

Elle est articulée en deux plans. Le modèle abstrait qui est une vue de haut niveau montrant les objectifs à la manière d'un diagramme des cas d'utilisation d'UML. Et les modèles descriptifs :

- le modèle de rôles montre les échanges entre les rôles internes et externes.

- le modèle de procédures est une vue intermédiaire qui détaille les processus définis dans le modèle abstrait. Il est très proche d'un diagramme de collaboration.

- le modèle d'opérations détaille un processus en fonction des rôles et du temps. Le formalisme en couloirs s'apparente aux diagrammes d'activités et aux BPD. La distinction entre les rôles internes et externes y est faite également. C'est une notion qu'on ne retrouve que dans OSSAD et qui pourtant a sa raison d'exister. En effet, on peut très rarement détailler autant la partie d'un processus s'effectuant en dehors de l'organisation qu'en interne. Une autre possibilité très intéressante permet de représenter des 'outils' c'est-à-dire des moyens organisationnels servant à la réalisation d'une opération.

- le modèle d'unité organisationnelle qui montre la structure hiérarchique avec les acteurs nommés donc réels et leurs rôles. C'est une avancée indéniable qui permet de faire le lien entre les processus et les ressources humaines de l'organisation.

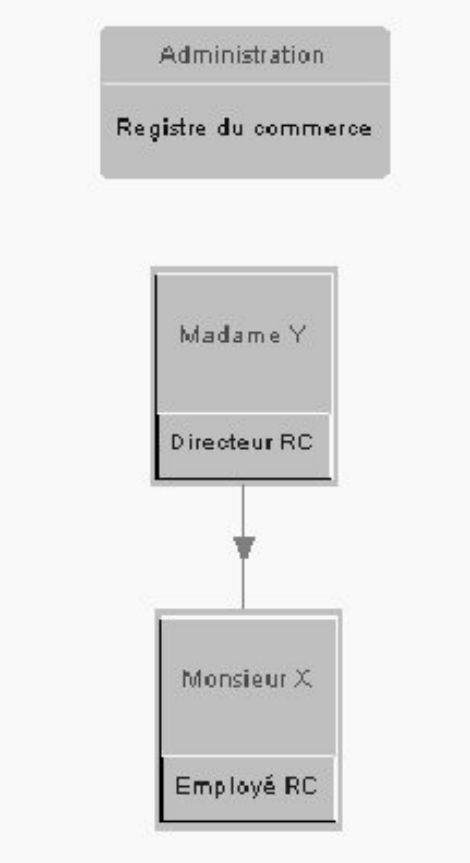


Figure I.13 Modèle d'unité organisationnelle.

Ce modèle nous intéresse tout particulièrement pour notre simulation puisqu'il nous donnera des informations sur le nombre d'acteurs et leur répartition précise sur les rôles avec parfois des acteurs qui en ont plusieurs.

1.3.2. ADONIS.

Il s'agit ici d'une méthode propriétaire crée par la société BOC (Business Objects Consulting : http://www.boc-eu.com/). Elle propose un logiciel qui la reprend et qui permet de modéliser et de simuler des processus.

Elle dispose de trois types de modèles :

- la carte des processus qui est la vue de haut niveau d'abstraction.

- le modèle de processus opérationnel qui est le détail et qui se présente encore sous la forme de couloirs d'activités. Comme dans OSSAD on peut y matérialiser des ressources quelles soient de type données ou organisationnelles.

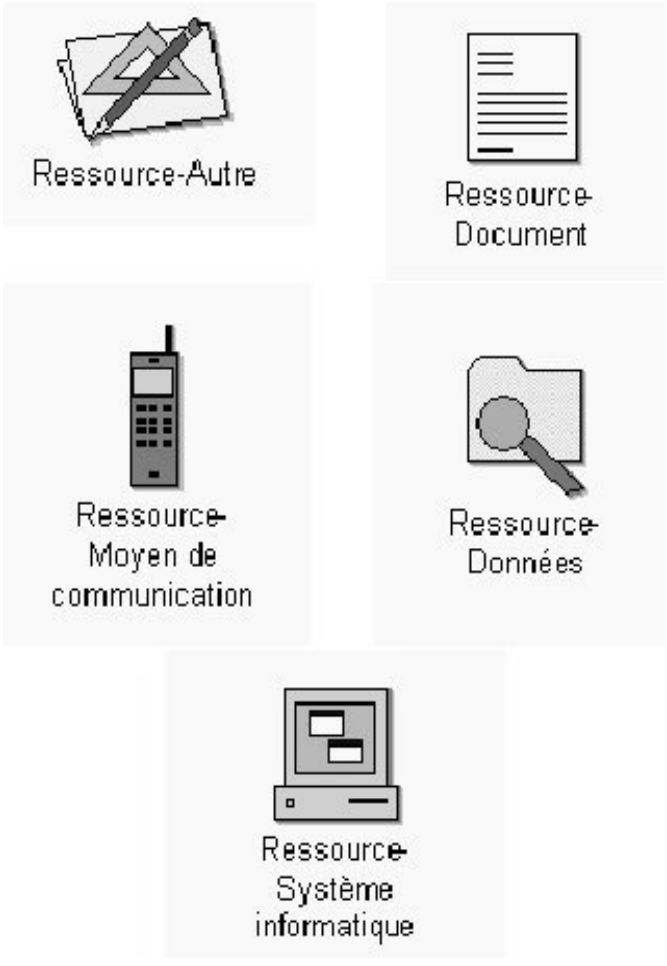


Figure I.14 Les types de ressources d'ADONIS.

- le modèle d'environnement de travail modélise très précisément la structure d'une organisation en terme de personnes, de rôles et de hiérarchies. Ce modèle peut être complété

par les mêmes informations de ressources que les précédentes pour les rattacher à un rôle ou à une unité organisationnelle.

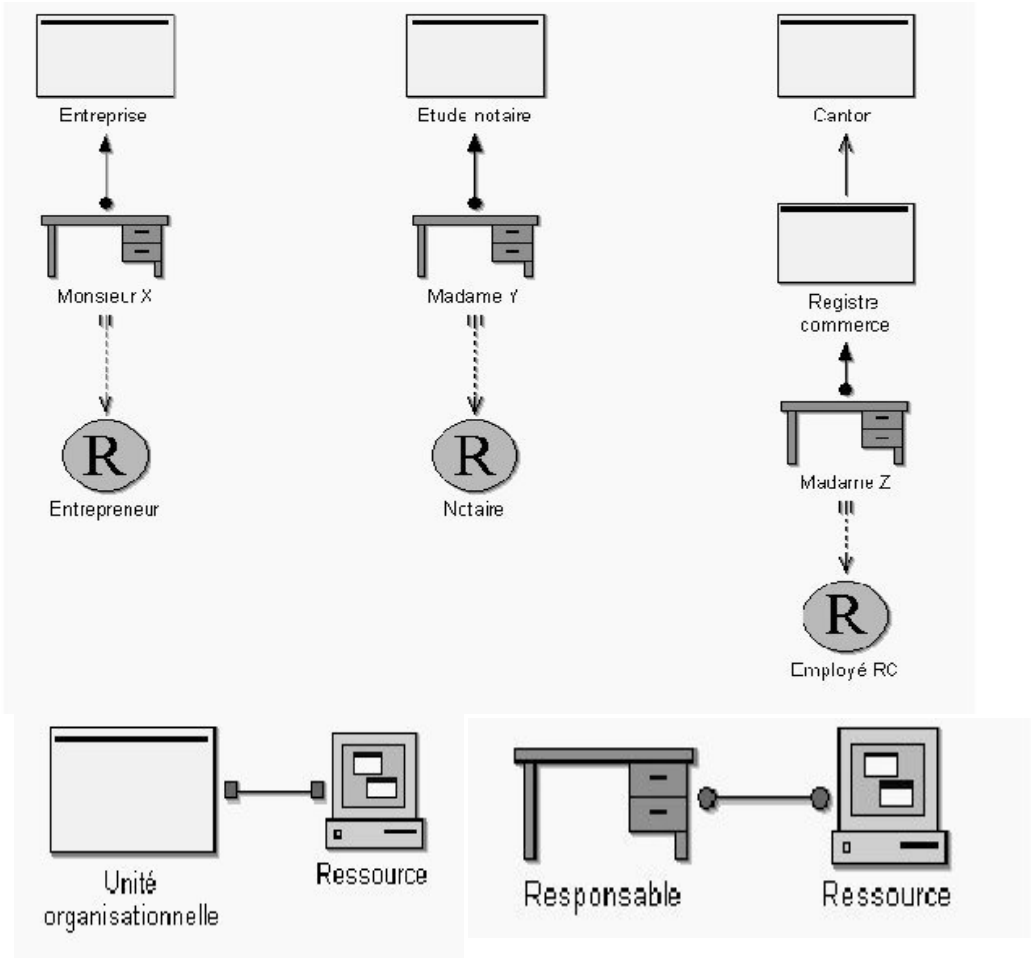


Figure I.15 Modèles d'environnement de travail.

Ce modèle organisationnel d'Adonis vient combler un manque crucial chez les autres notations. Il permet de décrire l'organisation réelle dans laquelle le processus se déroule. Il répond parfaitement à la question du 'Qui avec Quoi'.

La liaison avec le plan opérationnel s'effectue par le biais des rôles. Le modèle d'environnement de travail d'Adonis nous donne la vision des acteurs réels qui endossent le rôle. Il situe ces acteurs dans leurs unités organisationnelles respectives. Ainsi par cette décomposition en rôles, acteurs et unités organisationnelles, il devient aisé de spécifier quelles ressources sont disponibles à tous les niveaux de l'organisation.

Ce n'est pas si étonnant qu'Adonis dispose d'un tel modèle. En effet, comme nous

l'avons dit lors de sa présentation, la société BOC propose un simulateur des processus modélisés par cette méthode. Or pour pouvoir simuler, un tant soit peu correctement des processus métiers, il faut une modélisation de l'organisationnel tenant compte des acteurs et des ressources mises à leur disposition.

1.3.3. Conclusion sur OSSAD et ADONIS.

Ces deux notations fournissent des exemples de modélisations organisationnelles très développées. Elles nous ont appris que le plan organisationnel est d'une importance toute aussi grande que l'opérationnel dans les processus métier. Leur étude nous montre ce qu'il est nécessaire de modéliser sur ce plan pour compléter correctement la partie opérationnelle.

Le modèle d'environnement de travail d'ADONIS est le plus abouti. Il regroupe sur un seul modèle les unités organisationnelles, les acteurs, les ressources et les rôles. Il définit clairement les relations qui existent entre ces entités. Il complète parfaitement la vue opérationnelle en donnant des informations telles que, par exemple, le nombre d'acteurs pour un rôle, les acteurs à rôles multiples, les ressources dédiées ou partagées, etc.

D'OSSAD on retiendra la distinction entre les rôles internes et externes à l'organisation. Cette notion est importante car elle induit qu'un rôle externe ne se traite pas comme un interne puisque l'expert métier n'en a qu'une vision partielle.

1.4. Conclusion sur les notations diagrammatiques.

De cette étude des principales notations permettant la modélisation de processus, nous pouvons déduire une solution qui répond aux besoins de notre projet.

Nous pensons qu'il est nécessaire d'avoir deux vues distinctes, une opérationnelle et une organisationnelle. Nous utiliserons les notions de rôles et de ressources pour établir la liaison entre les deux modèles.

1.4.1. Le modèle opérationnel.

En ce qui concerne l'opérationnel, il est indéniable que BPMN s'impose devant UML par sa relative simplicité et son fort pouvoir d'expression. Comme nous l'avons déjà mentionné, son modèle unique (le BPD) est un atout primordial pour un travail efficace avec les experts métiers : ils savent généralement décrire leurs processus en disant comment cela se passe. De plus, en tant que standard de notation dédié aux processus métier, elle apporte toute la sémantique spécialisée à ce domaine.

Elle reprend un formalisme très ancien et répandu, celui des organigrammes (ou des 'flowcharts' de J. Von Neumann, A. Buks et H. Goldstine créés en 1946), dont on retrouve les grandes lignes (activités, transition, décisions,...) dans chaque notation. De part sa simplicité, ceux qui n'y seraient pas déjà habitués pourront l'acquérir rapidement.

Il faut prévoir d'étendre la notation BPMN qui ne dispose pas de la possibilité de modéliser l'utilisation de ressources. En effet, bien que les ressources appartiennent à la vue organisationnelle, la vue opérationnelle doit donner les activités qui les nécessitent. Le plus pratique, à notre avis, est de les concevoir comme des objets symboliques et de les relier aux activités qui les utilisent par un lien de type association.

1.4.2. Le modèle organisationnel.

Pour la vue organisationnelle, c'est à dire celle qui concerne les processeurs (hommes et machines) des processus métiers, nous préconisons un modèle inspiré de l'environnement de travail d'ADONIS. Nous pensons qu'il est nécessaire d'y ajouter la notion d'acteurs internes

et externes comme dans OSSAD. En effet, les rôles externes ne pouvant être aussi détaillés en qualité et en nombre que les rôles internes, la nature de leurs agents au sein notre plate-forme de simulation sera différente.

C'est une vue de cette nature qui nous permettra de concevoir notre représentation des processus à base d'agents logiciels. Ces derniers simulant des acteurs réels, elle permettra de leur attribuer le ou les rôles qui leurs incombent et, par conséquent, les parties de processus dont ils ont la responsabilité.

Les ressources seront également simulées par des agents logiciels. Ce modèle donnera individuellement leurs appartenances aux différentes entités organisationnelles. Nous respecterons, de la sorte, leur utilisation réelle par les acteurs au sein de l'entreprise.

2. LES ENRICHISSEMENTS DE LA NOTATION.

Nous avons au chapitre précédent déterminé une notation qui sied bien à notre projet par l'étude des principales méthodes en matière de modélisation de processus. Elle va nous permettre de créer des modèles qui sont une abstraction de la réalité. Ce sont des ensembles de signes et de textes qui vont permettre à l'utilisateur par interprétation de se faire une représentation mentale du processus.

On peut alors se demander si une simulation, c'est-à-dire une animation du modèle proche de la réalité, peut être directement obtenue. De la même manière que l'utilisateur à besoin d'interpréter, donc d'utiliser des informations supplémentaires qui lui viennent de sa propre expérience, le modèle à besoin d'être enrichi pour obtenir une simulation.

Nous allons détailler maintenant quels sont les enrichissements nécessaires à nos modèles opérationnels et organisationnels dans l'optique de créer des simulations de processus métier telles qu'elles sont définies dans le projet. En particulier ceux qui permettront de rendre les modèles exécutables et animés graphiquement par l'utilisation d'un système multi-agents.

2.1. Les enrichissements du modèle opérationnel.

2.1.1. Les instances d'exécutions de processus.

Un processus métier est constitué de l'ensemble des activités internes d'un métier dont l'objectif est de fournir un résultat observable et mesurable. Lorsqu'on dispose du modèle opérationnel, on est en possession d'une vue abstraite montrant l'ensemble des possibilités offertes à chaque exécution en terme d'opérations à effectuer.

Pour animer des exécutions de processus, il faut donc générer des instances de processus ayant chacune un parcours et une durée de vie qui lui sont propres. Il est nécessaire qu'elles aient une certaine consistance au sein de notre système pour pouvoir les faire s'exécuter mais surtout les représenter graphiquement puisque c'est l'objectif principal. Il faut également les différencier par des notions de volumétrie et de priorité. En effet, ces critères permettront de voir se manifester les phénomènes particuliers que nous recherchons tels que les sur ou sous occupations d'acteurs, les états des files d'attentes des instances d'exécutions à certains stades,...

En fonction du type de plate-forme logicielle qui servira à la simulation, on pourra utiliser un système à jetons circulant entre les activités ou mettre en place une gestion en multithreading.

Il arrive que plusieurs instances d'exécutions d'un même processus se déroulent en même temps dans la partie sous la responsabilité d'un acteur. C'est le cas, par exemple, quand une instance d'exécution est en attente d'une réponse, l'acteur utilise ce temps d'attente pour en traiter une autre. Il sera nécessaire de mettre en place une gestion 'multi-instances' avec des règles de répartition du temps (limité pour cet acteur) pour refléter le comportement attendu.

2.1.2. L'événement déclencheur.

On sait qu'un processus a toujours un début et un seul. Qu'il est pratiquement toujours déclenché par un événement extérieur (sauf dans certains cas de règles métier comme stock< 10 unités). C'est donc la survenue de cet événement déclencheur qui créé une instance de

processus.

En fonction de sa nature, la simulation ne sera pas la même :

- pour les événements temporels, il suffit qu'ils soient bien spécifiés ('tous les lundis', 'toutes les 5mn',...) et l'on pourra déclencher le processus à l'aide de l'horloge du système.

- pour les événements de type 'règle' les formes peuvent être diverses avec parfois des déclenchements que l'on ne peut guère prévoir. Il faudra les simuler soit statistiquement soit par le déclenchement direct de l'utilisateur de la plate-forme.

- pour les événements correspondant à l'arrivée d'un message, il faut enrichir le modèle par une donnée statistique relative à sa survenue (un nombre d'événements pour un temps donné). Selon les cas ce peut être une quantité fixe ou variable.

2.1.3. Le parcours et la durée.

Pour ce qui est du parcours de l'instance, cela dépend du chemin pris à chaque structure de contrôle rencontrée. Il faut donc ajouter à chaque point de décision du modèle des informations statistiques, fournies par l'expert métier, donnant la répartition sur les différents chemins possibles.

En ce qui concerne la durée d'exécution, il est obtenu par le cumul des temps de réalisation de l'ensemble des tâches effectuées sur le parcours de l'instance, mais également par l'ajout d'éventuelles attentes. Pour le temps d'exécution d'une tâche, là encore, c'est l'expert métier qui peut donner une estimation qu'il faudra rajouter au modèle. Ce peut être soit un temps fixe soit un temps qui va dépendre de la volumétrie à traiter.

Les délais d'attentes peuvent être de plusieurs natures. Il y a tout d'abord l'attente entre la survenue de l'événement déclencheur et sa prise en charge par un des acteurs du processus. Il y a aussi les attentes dues aux synchronisations ou à l'indisponibilité de ressources organisationnelles.

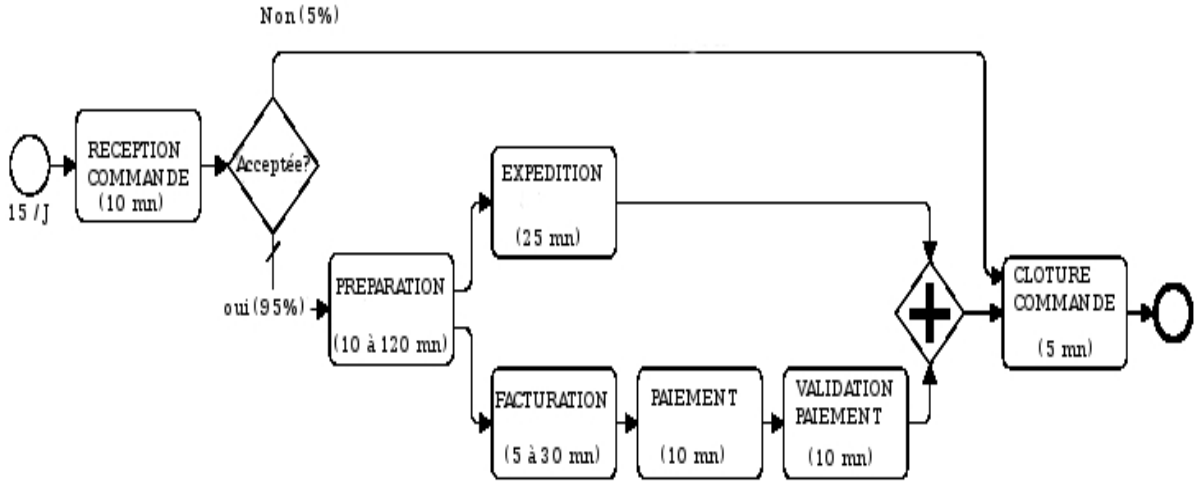


Figure II.1 Processus commande annoté.

La figure ci-dessus montre un exemple d'annotations nécessaires à la simulation dans le cas du traitement d'une commande. On s'aperçoit que certaines tâches on des durées fixes (réception, paiement,...) et d'autres des durées variables (préparation, facturation) car elles sont proportionnelles à la volumétrie de la commande.

Le test 'acceptée?' doit être pourvu d'une répartition statistique donnée par l'expert de manière à se rapprocher le plus possible de la réalité lors de la simulation.

L'événement déclencheur à une indication de cadence qui permettra de créer des instances d'exécutions distinctes de ce processus.

2.1.4. De la globalité à l'individualité.

Le modèle opérationnel donne une vision du processus métier tel que l'expert le conçoit, c'est-à-dire dans sa globalité. Nous avons donc sur un même graphe le comportement des différents rôles impliqués.

Si nous souhaitons simuler le processus à l'aide d'un système multi-agent où chaque agent simule le comportement d'un ou plusieurs rôles précis, il faut pouvoir extraire du modèle opérationnel la ou les parties qui le concerne et qui se matérialisent par un ou plusieurs couloirs d'activité.

Mais, afin de ne pas perdre d'information, il faut annoter chaque connecteur entrant ou

sortant franchissant la frontière du couloir d'activité. Ceci est nécessaire pour pouvoir refaire les liens entre les différents acteurs lors de l'exécution du processus par la plate-forme de simulation.

Nous allons appliquer cette transformation sur l'exemple suivant :

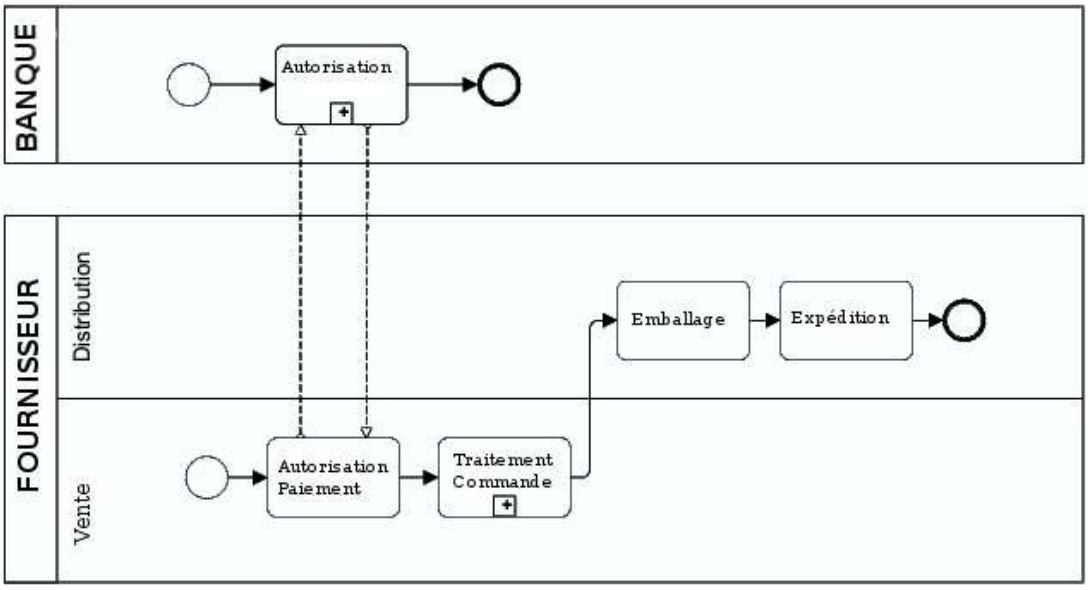


Figure II.2 Modèle opérationnel global.

Nous avons sur la figure précédente un processus avec deux pools (Banque et Fournisseur) ce dernier ayant deux couloirs (vente et distribution). Pour extraire le comportement du rôle 'vente' et le rendre apte à la simulation par un agent logiciel, nous devons indiquer vers quelles entités vont les deux flux messages et le flux séquence qui sortent du couloir.

Pour se faire, on peut conserver la sémantique de BPMN est noter les flux messages comme des événements de type message et matérialiser le flux séquence final par un événement lien comme illustré par la figure suivante.

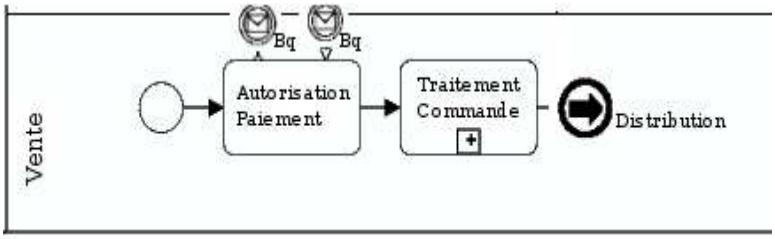


Figure II.2 Modèle opérationnel individualisé.

Ce lien terminal correspond à l'envoi d'un message vers le rôle indiqué. Il doit, selon les choix d'implémentation, véhiculer le jeton ou passer les paramètres de l'instance et terminer la thread. L'instance de processus, elle, n'est pas terminée. Elle va poursuivre son exécution chez un acteur de rôle 'Distribution'.

Dans les cas des messages intermédiaires l'instance reste bloquée sur la réception du message. L'émission va créer un flux en parallèle de la même instance.

Il convient de bien repérer le début et la ou les fins du processus métier pour que la plate-forme génère et détruise les instances.

2.2. Les enrichissements du modèle organisationnel.

Ici encore, ce sont des données quantitatives qui vont nous manquer. Nous ne traiterons en détail que de l'organisation interne puisque dans la plus grande majorité des cas l'organisation des rôles externes échappe à l'expert d'un processus métier.

Au niveau interne, nous avons surtout besoin d'avoir des informations sur les acteurs. Il nous faut savoir, par exemple, s'ils sont employés à temps plein ou autrement, la répartition de leur temps de travail en cas de rôles multiples, s'il existe un système d'établi de priorité en cas de concurrence de plusieurs acteurs sur le même rôle,...

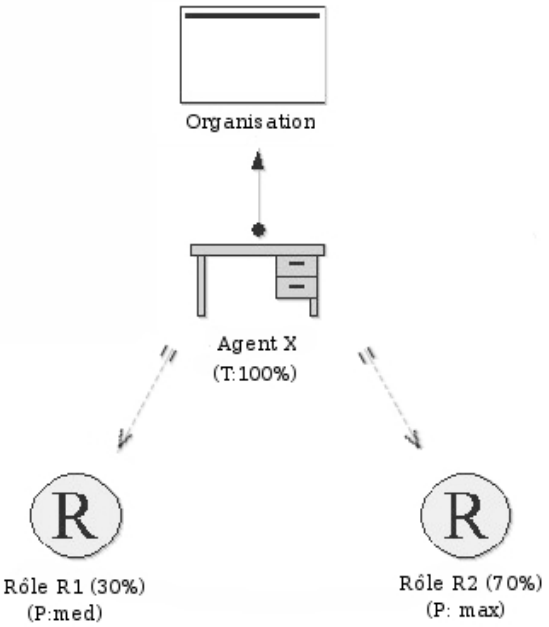


Figure II.3 Exemple de Modèle Organisationnel annoté.

La figure II.3 nous montre un exemple d'annotation possible d'un modèle organisationnel. On y a représenté un agent X employé à temps plein (T:100%) jouant deux rôles : le rôle R1 pour 30% de son temps et le rôle R2 pour 70%. Cet agent a une priorité maximum sur le rôle R2, ce qui lui confère le privilège de prendre en charge une instance pour laquelle il est en concurrence avec un autre agent jouant le même rôle.

Les rôles externes, quant à eux, seront traités dans leur globalité. Généralement, il s'agit de rôles génériques pour lesquels une individualisation est difficile ou trop coûteuse en ressources. C'est le cas pour des rôles tels que : client(s), fournisseur(s), banque(s),... Nous choisirons donc de les représenter par un symbole unique qui est équivalent, du point de vue de l'abstraction, aux étiquettes en tête des couloirs d'activités.

3. UN STYLE ARCHITECTURAL ADAPTATIF.

Le projet de représentation interactive des processus métier doit permettre la prise en compte dynamique de tout changement souhaité par l'expert. C'est-à-dire, que l'on doit être capable d'apporter des modifications tant opérationnelles qu'organisationnelles en cours de simulation.

La nécessité d'une architecture logicielle à mettre en oeuvre suffisamment flexible, nous a amené à étudier un style architectural répondant à ces attentes : les AOMs : Adaptive Object-Models.

3.1. LES "ADAPTIVE OBJECT-MODEL"

Le concept des "Adaptive Object-Model" (AOM) prend ses racines chez Ralph E. Johnson en 1998 dans un article qu'il intitule "*Dynamic Object Model*" [Joh98]. Il y relate un style architectural nouveau qu'il a remarqué dans plusieurs démonstrations au OOPSLA'97 (*Object-Oriented Programming, Systems, Languages, and Applications* : conférence annuelle sur l'Objet de l'*Association for Computing Machine*).

Ce style d'architecture est utilisé pour rendre extrêmement extensible un système, à tel point, qu'il peut même être étendu lors de son exécution par des non-programmeurs si les Interfaces Homme-Machine (IHM) idoines sont à disposition. C'est une architecture qui est composée de plusieurs patterns, dont certains proviennent directement de l'ouvrage de référence dans ce domaine : "*Design Patterns: Elements of Reusable Object-Oriented Software*" [GHJV95] du célèbre 'clan des quatre' dont Johnson est un membre. C'est lui qui change, plus tard, la dénomination en AOM et créé une équipe de recherche sur ce thème à l'université de l'Illinois.

Le principe des AOM est de représenter les classes, les attributs, les relations et les comportements comme des méta-données d'un domaine. Pour ce faire, le modèle Objet est conservé au sein d'une base de données. La spécification des classes réelles se fait dans un sous-niveau au précédent par interprétation à l'exécution de ces méta-données qui peuvent être combinées et affinées. De cette manière, une collection d'instances spécifiques d'une classe fortement générique est substituée à la collection de classes spécifiques produite par le mécanisme classique de l'héritage. On distingue donc 2 niveaux d'abstraction connus sous les dénominations très explicites de Knowledge et Operational Level chez Fowler [Fow97]. Cela permet par la simple modification des méta-données de répercuter sur le niveau opérationnel, sans arrêter le système, les changements souhaités.

Voyons maintenant plus en détail ce style architectural. Nous verrons tout d'abord le pattern *TypeObject* [JW98] qui permet la création des deux niveaux d'abstraction en séparant les entités du niveau opérationnel de leurs 'entités-types' situées au niveau connaissance. Puis

le pattern *Property* [FY98] permettant l'ajout dynamique d'attributs aux classes. Enfin le comportement sera défini à l'aide du pattern *Strategy* [GHJV95].

3.1.1. Le pattern "*TYPEOBJECT*"

En programmation orientée Objet traditionnelle, on utilise le mécanisme d'héritage de classe pour créer de nouveau type d'objets dont la nature est proche d'une définition de classe déjà existante. Or, si l'on peut créer dynamiquement une nouvelle instance, on ne peut pas faire de même avec une classe. Si l'on souhaite étendre un système on doit écrire le code de la nouvelle classe et recompiler.

Dans les domaines où les changements sont fréquents, les développeurs sont très souvent amenés à sous-classer maintes et maintes fois les mêmes classes avec les inconvénients qui en résulte (notamment au niveau de la maintenance et la documentation). Le pattern "*TypeObject*" répond à cette problématique en utilisant un jeu de deux classes qui va venir remplacer l'ensemble que forme la classe et ses multiples sous-classes.

Une première classe sert de 'type' à l'autre. Elle comprend tous les attributs et méthodes partagés par l'ensemble des instances du même 'type'. La seconde, qui est la classe des instances opérationnelles du système, contient un pointeur qui permet la mise en relation de l'instance avec son 'type'.

On remarque que ce pattern n'est une solution que dans le cas où on sait prédire sur quels éléments les changements vont avoir lieu.

Nous trouvons cette relation entre deux classes mal désignée par 'type' qui porte à confusion avec la notion de type en orienté objet (type de donnée ou d'objet). D'ailleurs, on retrouve déjà ce pattern en 1992 chez Peter Coad sous le nom 'Item Descriptor' [Coa92] ou en 1995 avec les 'PowerType' [MO95] de James Odell qui sont repris par UML.

Prenons un exemple concret celui des livres d'une bibliothèque publique. La façon de traiter ce problème qui vient tout de suite à l'esprit et de représenter chaque livre par une instance d'une classe Livre. Mais, il y a beaucoup d'exemplaires de la même oeuvre ce qui

apporte ainsi de la redondance d'information. Utilisons donc le mécanisme de l'héritage et faisons des sous-classes de la classe Livre contenant les informations relatives à chaque oeuvre. Nous avons ainsi résolu le problème de la redondance d'un titre, d'un auteur, d'un éditeur,... Nous avons maintenant un modèle objet avec une classe abstraite Livre et une multitude de sous-classes (*Figure III.1*).

Mais le nombre de sous-classes à créer risque d'être beaucoup trop important. De plus, l'arrivée d'un livre n'ayant pas sa sous-classe va nécessiter l'ajout du code de la nouvelle classe et par conséquent une recompilation du système.

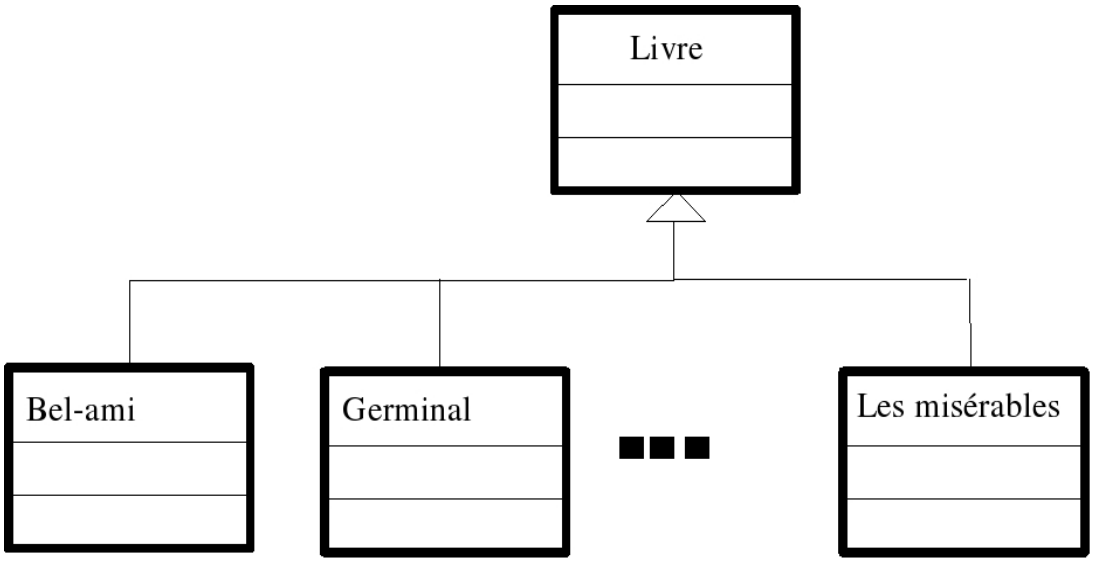


Figure III.1 Modèle Objet 'classique' des livres d'une bibliothèque.

L'application du pattern "TypeObject" à cet exemple se traduit par la création de deux classes. La classe type se matérialise en la classe Oeuvre contenant des informations comme le titre, l'auteur,... Et la classe Livre classe dont héritent les instances du système et qui contient, en plus de ses attributs de valeurs spécifiques (la cote, prêté,...), un attribut oeuvre servant de pointeur vers le type de l'instance (*Figure III.2*).

A l'exécution, ce modèle objet permet de créer autant d'instances de Livre qu'il y a dans la bibliothèque et autant d'instances d'œuvres que nécessaires. Si un livre d'une oeuvre nouvelle est ajoutée à la bibliothèque, il suffira d'abord de créer une nouvelle instance de la

classe Oeuvre avec les attributs qui y correspondent. Puis de créer une instance de Livre pointant sur l'instance d'Oeuvre précédemment créée. On pourra, par la suite, rattacher d'autres livres partageant la même oeuvre à cette instance.

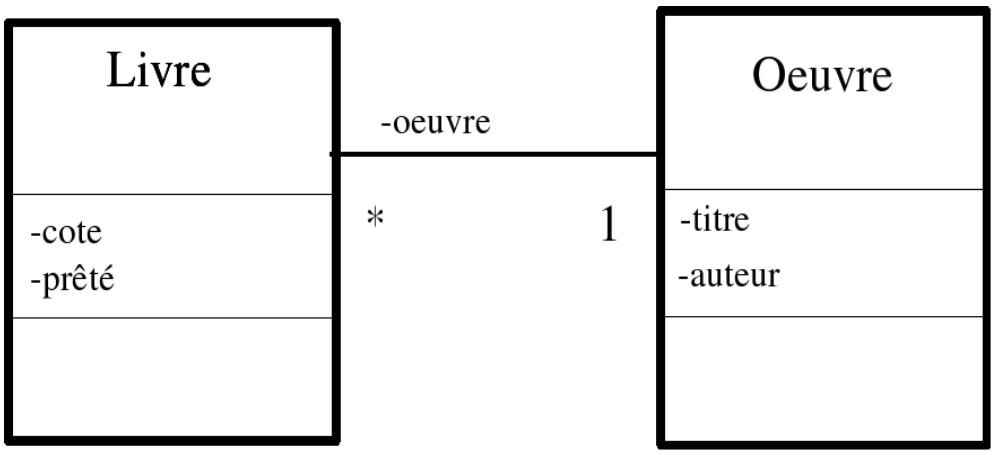


Figure III.2 Modèle Objet des livres avec le pattern "TypeObject".

L'utilisation du pattern "TypeObject" pose un problème au niveau des attributs. Dans le modèle classique de la Figure III.1, le recours aux sous-classes permet également de faire varier les attributs en nature et en nombre. Dans les AOMs ce problème est résolu par l'utilisation du pattern "Property" [FY98].

3.1.2. Le pattern "*PROPERTY*"

Le principe de ce pattern est de créer une instance variable contenant une collection d'attributs. Ainsi chaque instance à une sorte de dictionnaire qui relie un nom d'attribut à sa valeur. Il faut cependant prévoir des méthodes permettant la gestion de ce dictionnaire (ajout, affectation, suppression,...). Ce mode d'implémentation des valeurs d'attributs d'une instance a était décrit pour la première fois par Beck sous le nom de "Variable State Pattern"[Bec96].

Mieux encore, une classe Property peut être créer pour détenir le nom, la valeur mais également le type (*Figure III.3*). Cela est particulièrement nécessaire dans le cas de la création

d'interfaces entre l'utilisateur et le système agissant sur ces attributs afin d'éviter des saisies incongrues. L'interface pourra faire la vérification que la valeur entrée est bien du type attendu.

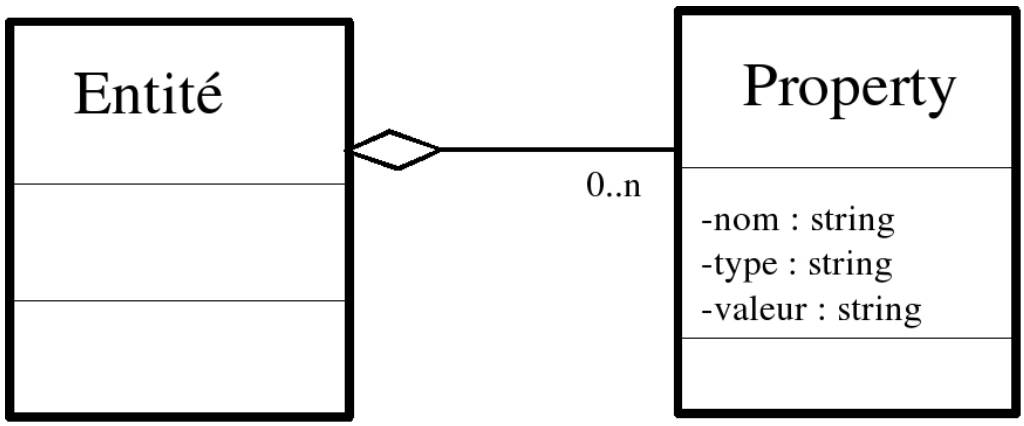


Figure III.3 Le pattern "Property".

Si on applique maintenant le pattern "TypeObject" sur la classe Entité on s'aperçoit que les attributs génériques relégués dans la classe EntitéType ne pourrons pas bénéficier de la même flexibilité que ceux implémentés par le pattern "Property".

Johnson et son équipe vont avoir l'idée d'appliquer le pattern "TypeObject" également sur la classe Property. Ils vont ainsi créer une nouvelle classe au niveau connaissance, la classe PropertyType (*Figure III.4)*. Elle va permettre la création d'instances qui sont des types d'attributs qui serviront dans la classe Property mais aussi dans la classe EntitéType par le truchement d'une seconde application (au niveau connaissance) du pattern "Property". Le modèle ainsi obtenu a été baptisé "TypeSquare" en raison de sa forme.

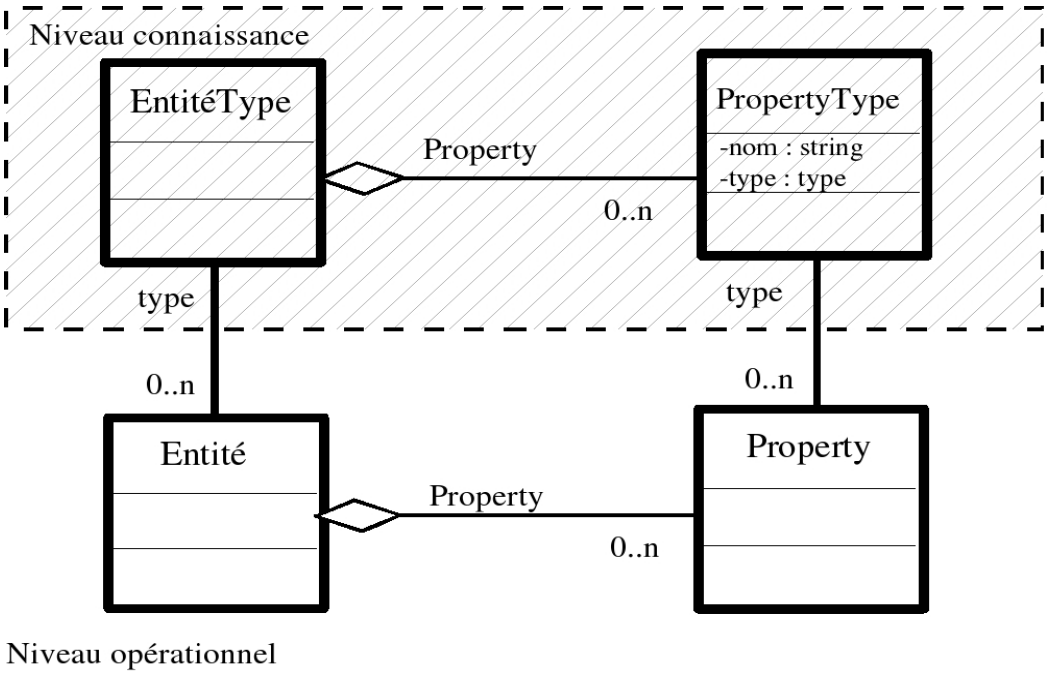


Figure III.4 le modèle "TypeSquare"

Nous venons maintenant d'établir un modèle objet à quatre classes qui permet de définir et de modifier dynamiquement sans arrêter le système de nouvelles entités. Dans l'état actuel, ces entités ne varieront que par leurs attributs. Ce modèle ne se préoccupe pas du comportement des entités créées. Si on en reste là, il faudra avoir recours à l'héritage pour modifier le comportement.

La prochaine étape consiste donc en l'extension de ce modèle pour permettre de doter les entités d'un comportement flexible à l'exécution.

3.1.3. Le pattern "*STRATEGY*"

Le patron "Strategy" [GHJV95] cherche principalement à séparer un objet de ses comportements (ou algorithmes) en encapsulant ces derniers dans des classes. Pour ce faire, on doit alors définir une famille de comportements ou d'algorithmes encapsulés et interchangeables. Cette famille doit partager une interface commune qui se matérialise en la classe abstraite Strategy (*Figure III.5*). Cette interface permet d'avoir les paramètres nécessaires aux différentes variantes des algorithmes. Mais aussi de gérer les cas où des paramètres non-nécessaires sont passés.

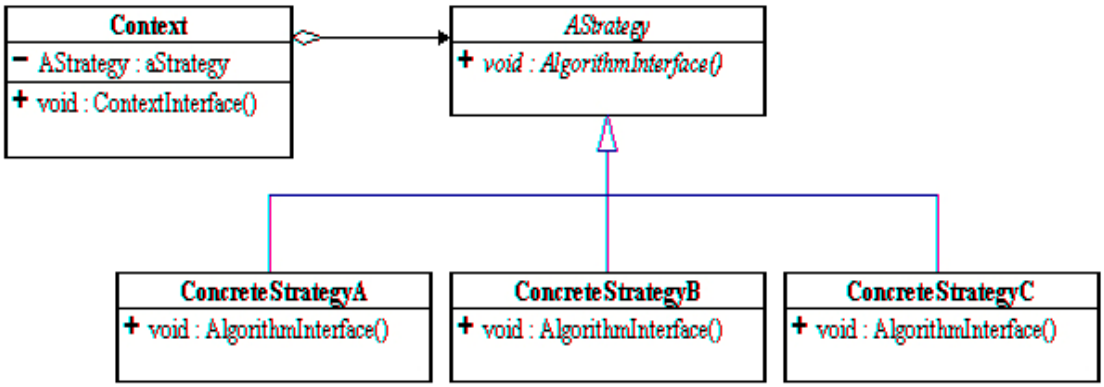


Figure III.5 Le pattern "Strategy"

Ce pattern ne permet, cependant, que de manipuler des comportements définis uniquement lors de la phase de conception donc avant le déploiement de l'application. On ne peut alors pas intervenir sur ces algorithmes à l'exécution. Pour y arriver, on doit ajouter un niveau supérieur, un méta-niveau, qui permettra d'intervenir sur ces comportements. On va utiliser la technique de la réflexion qui va permettre au système d'avoir une représentation de son domaine et qui lui autorise des modifications.

Pour ce faire, le pattern "Composite" va être combiné au pattern "Strategy".

3.1.4. Le pattern "*COMPOSITE*"

Ce pattern forme une structure arborescente d'un ensemble de composants. Il permet à la classe cliente de composer et d'utiliser une hiérarchie complexe d'objets. Il est constitué de trois éléments :

- La classe Composant est une classe abstraite qui détient l'interface commune à tous les objets de la composition. Elle comporte les opérations de gestion de la composition (ajout et suppression de descendants). Les méthodes appelées sur les composites sont propagées sur leurs descendants jusqu'à atteindre la feuille qui l'exécutera.

-La classe Composite est la sous-classe des composants formés de plusieurs composants (ses fils).

-La classe Feuille est la sous-classe des composants qui ne possèdent pas de fils.

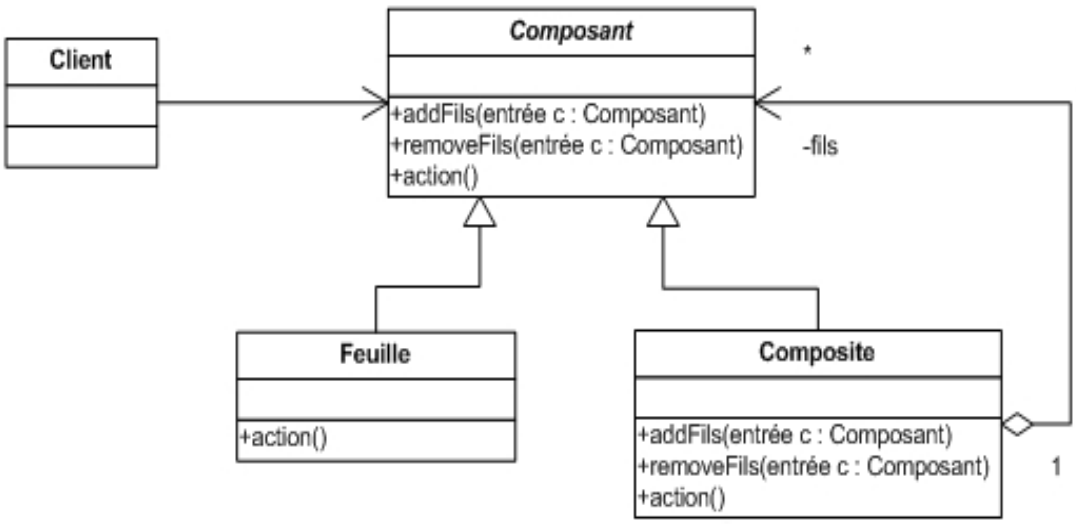


Figure III.6 Le pattern "Composite"

Dans les AOMs, l'adjonction du pattern "Composite" au pattern "Strategy" a pour but de permettre la création de comportements complexes qui sont en général soit des règles métier simples ou une composition de ces règles métier. Le motif de conception ainsi créé porte le nom de "RuleObjects"[Ars00] et vient compléter le modèle "TypeSquare" pour former ce que la littérature à coutume d'appeler "le coeur des AOMs"[Joh98,YBJ01, Raz01](*Figure III.7*).

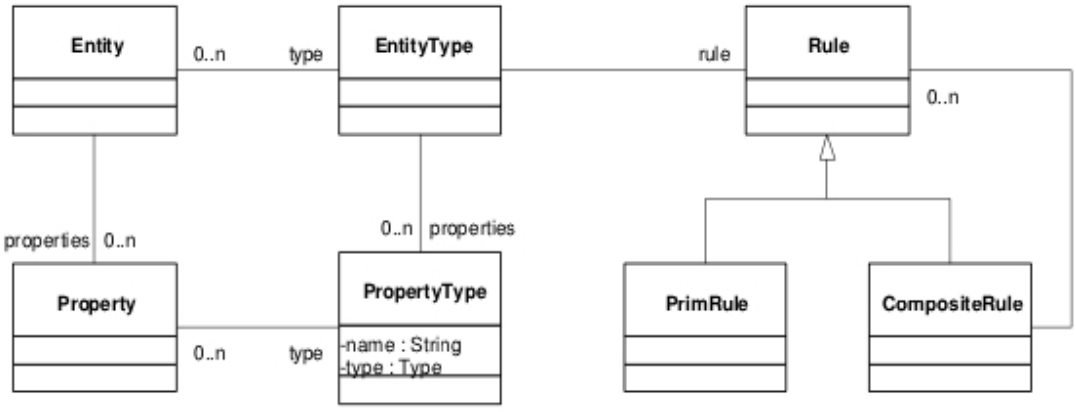


Figure III.7 le "cœur des AOMs"

En général les types d'entités, les propriétés et les comportements sont conservés au sein d'une base de données. Si cette dernière est de type 'orientée Objet', les objets du domaine seront directement instanciés à partir des méta-données (Types, propriétés et règles métier).

Sinon, les objets du domaine seront créés depuis une base de données classique en utilisant des fichiers XML par exemple. Cette opération nécessitera d'utiliser d'autres motifs de conception pour être menée à bien. On pourra utiliser :

- Builder : qui permet de créer des objets complexes par la spécification de types et de contenus. C'est une création en différentes étapes qui permet de créer des instances ayant des représentations différentes.

- Interpreter : qui permet de créer un langage simple de manipulation des objets règles avec des opérateurs (si..alors ; boucles ; ...)

3.2. APPLICATION DES AOMs AUX PROCESSUS METIER.

Nous avons précédemment développé le concept des AOMs qui permet de définir et de modifier des objets à l'exécution. Nous allons maintenant mettre cette technique en pratique sur un exemple simple de processus métier pour en juger les qualités et les défauts.

La plate-forme de représentation interactive étant basée sur un système multi-agents, nous ne parlerons plus en termes d'objets mais d'agents logiciels.

3.2.1. Le modèle AOM des processus métier.

Il nous faut tout d'abord adapter le modèle des AOMs à la problématique des processus métiers. Dans la plate-forme de simulation, nous aurons des agents logiciels simulant les acteurs réels des processus métier. Ils seront issus d'une classe AgentPM sous-classe de la classe Agent de la plate-forme multi-agents employée.

Chaque AgentPM sera associé par un lien «TypeObject» à son AgentPM_Type. Ce sont ces derniers qui feront le lien entre les AgentPM et leurs comportements. Ils seront porteurs de certains attributs : le rôle organisationnel notamment.

En ce qui concerne le comportement, nous allons transposer le pattern 'RuleObjects' des AOMs aux besoins des processus métier. L'adaptation principale consiste à définir les primitives nécessaires (tâche, envoi et réception de messages,...).

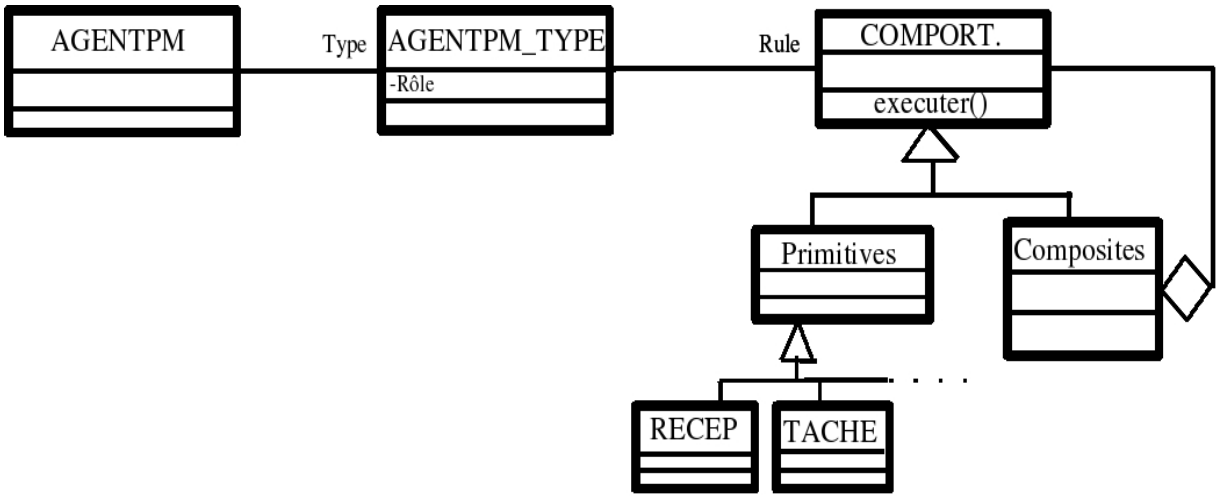


Figure III.8 le modèle AOM des processus métier.

Les primitives vont être déterminées à partir des éléments de la notation du modèle opérationnel que nous avons défini au chapitre II. Il s'agira alors de créer un ensemble de classes réalisant le comportement élémentaire de chacun de ces éléments (l'occupation de l'agent pour les tâches, la prise de décision pour les structures de contrôle, l'envoi d'un message à un autre agent,...). Elles permettront de créer les instances qui correctement paramétrées et combinées formeront le comportement des agents comme nous le détaillerons quand nous aborderons l'éditeur de comportements (cf. chapitre IV.3).

3.2.2. Exemple d'application.

La figure III.9 montre l'exemple que nous avons retenu (tiré de [Cru03]) et qui va nous servir à détailler le fonctionnement du modèle AOM des processus métier.

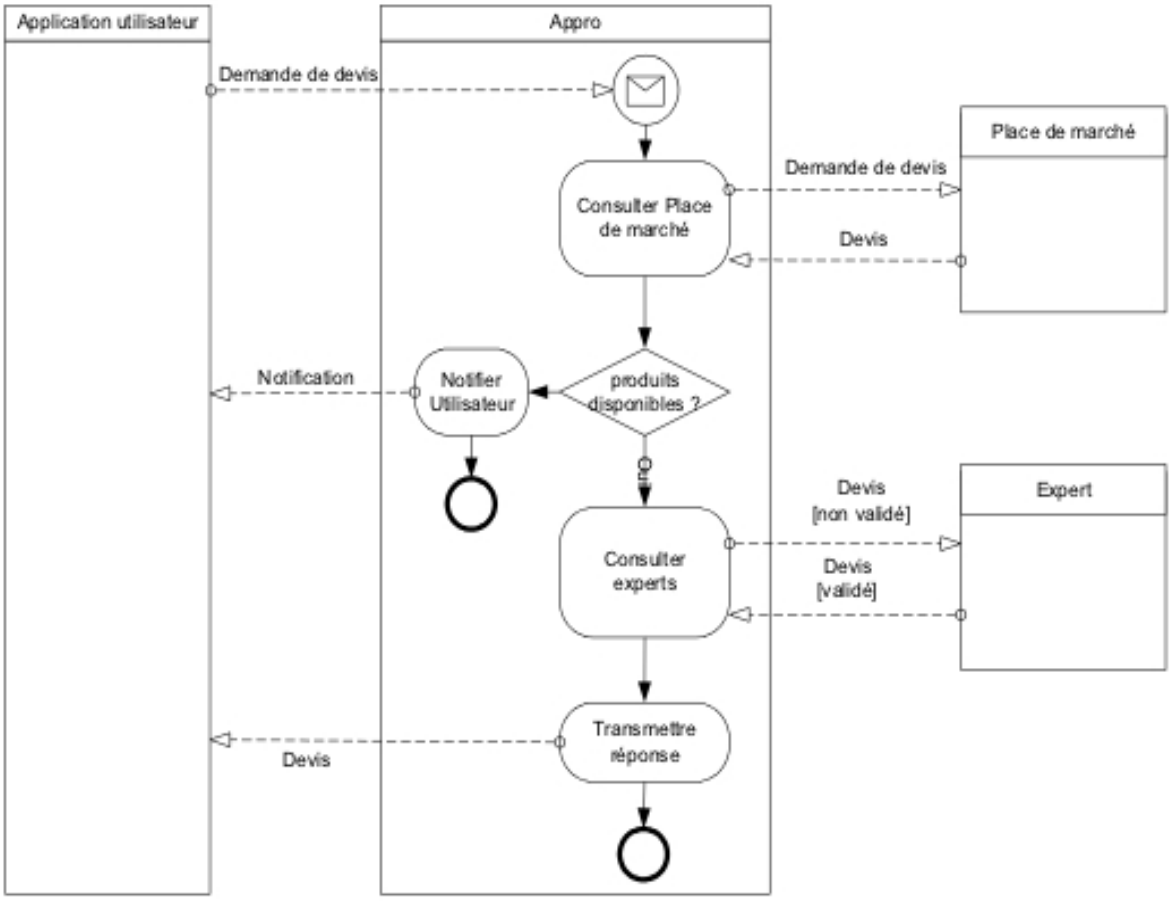


Figure III.9 le processus appro.

Cet exemple montre le comportement d'un rôle Appro et ses relations avec trois autres entités (Application Utilisateur, Place de marché et Expert). L'élément déclencheur est la réception d'une demande de devis par l'application utilisateur. Puis APPRO exécute un sous-processus de consultation de la place de marché. Dans le cas où les produits ne seraient pas disponibles il en avise l'application utilisateur. Sinon, après l'avoir fait valider auprès d'un expert, il transmet le devis.

La transposition de cet exemple dans notre modèle AOM des processus métier se concrétise par la création de trois agents principaux comme le montre la figure III.10.

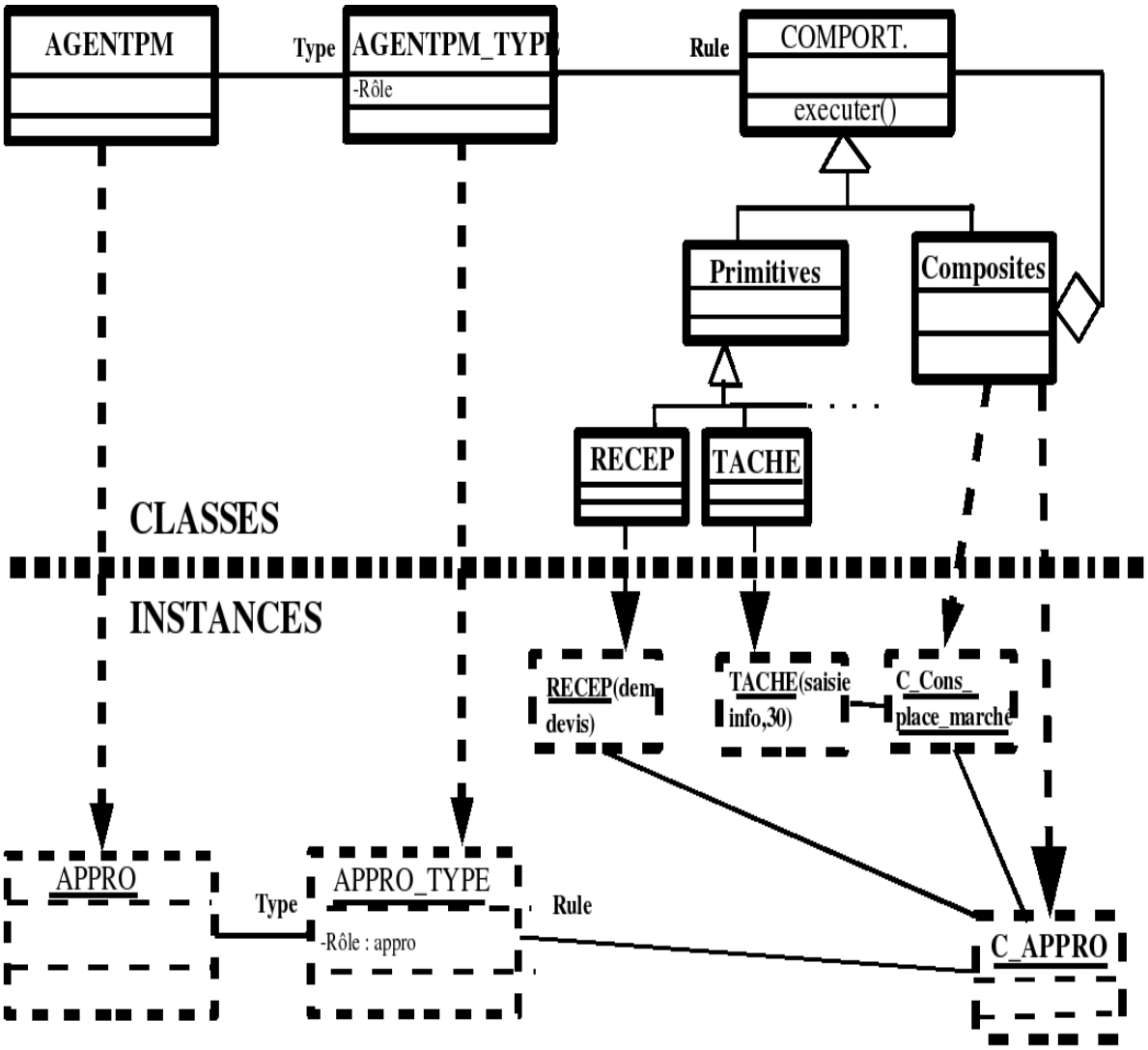


Figure III.10 Exemple d'instanciations pour l'agent APPRO.

L'agent logiciel APPRO qui dans la plate-forme simule le comportement d'un acteur endossant ce rôle est une instance de la classe AGENTPM. C'est par son lien 'Type' avec l'agent APPRO_TYPE qu'il se spécialise.

L'agent logiciel APPRO_TYPE est instance de la classe AGENTPM_TYPE. Sa fonction principale est de donner le comportement par l'intermédiaire du pattern RuleObject. Il fait également la correspondance entre le rôle et son comportement. Ainsi si l'on veut créer un second agent du même rôle il suffit de créer une instance de la classe AGENTPM et de la faire pointer sur APPRO_TYPE.

L'agent logiciel C_APPRO est une instance de la classe COMPORTEMENT. C'est un Composites qui est constitué de primitives et d'autres composites. Il est la traduction de la partie opérationnelle

Sur la figure III.10, nous avons représenté les premiers éléments de C_APPRO, il y a tout d'abord une instance RECEP(dem_devis) de la classe RECEP des primitives de réception de messages. L'élément suivant est un autre composite : C_Cons_place_marché puisqu'il s'agit du sous-processus 'consulter place de marché'. Ce dernier est lui-même constitué de primitives telles que :

- TACHE(saisie_info,30) qui est une activité élémentaire durant 30 minutes.
- ENV(demande_devis,place_marché).
- RECEP(devis).

Cet exemple nous montre la flexibilité de cette architecture. On remarque que si l'on veut modifier le comportement des acteurs d'un rôle il suffit de modifier le comportement attribué à l'AGENTPM_TYPE. Si l'on souhaite ajouter des acteurs à un rôle, il faut instancier un nouvel AGENTPM et le faire pointer sur l'agent type. Il est également possible de préparer plusieurs comportements pour un rôle et les substituer dynamiquement par la modification du lien 'Rule'.

Toutes ces modifications sont possibles puisqu'elles sont apportées à des objets et ne

nécessitent pas d'avoir à modifier les classes du système. Il est tout de même nécessaire que ces différents objets disposent des méthodes permettant d'effectuer toutes les modifications souhaitées par les utilisateurs.

Autrement dit, les classes AGENTPM, AGENTPM_TYPE et COMPORTEMENT doivent être créer en veillant à être les plus complètes possible au niveau des méthodes permettant les modifications dynamiques souhaitables de leurs instances. Les oublis, eux, ne pourront pas être ajoutés en cours d'utilisation de la plate-forme.

4. SPECIFICATIONS FONCTIONNELLES.

Dans les chapitres précédents nous avons procédé à l'étude des différents domaines nécessaires à la représentation interactive des processus métier.

Dans ce dernier chapitre, nous allons nous recentrer sur l'objet du projet en donnant les spécifications fonctionnelles de la plate-forme de représentation interactive de processus métier.

Nous prendrons comme document de référence le "Contrat de recherche externalisée : FT/R&D-GREYC : Représentation interactive de processus métier" (en Annexe 1).

4.1. COMPORTEMENT DE LA PLATE-FORME.

Le projet de Représentation Interactive de Processus Métier (RIPM) à pour but de créer une plate-forme permettant aux responsables de processus métier de pouvoir visualiser le comportement des instances de leurs processus se déroulant en parallèle. Mais également, de pouvoir effectuer dynamiquement des modifications et ainsi voir les répercussions qu'elles entraînent.

Pour bien concevoir ce que les utilisateurs attendent de cette plate-forme, nous avons proposé le diagramme des cas d'utilisations suivant.

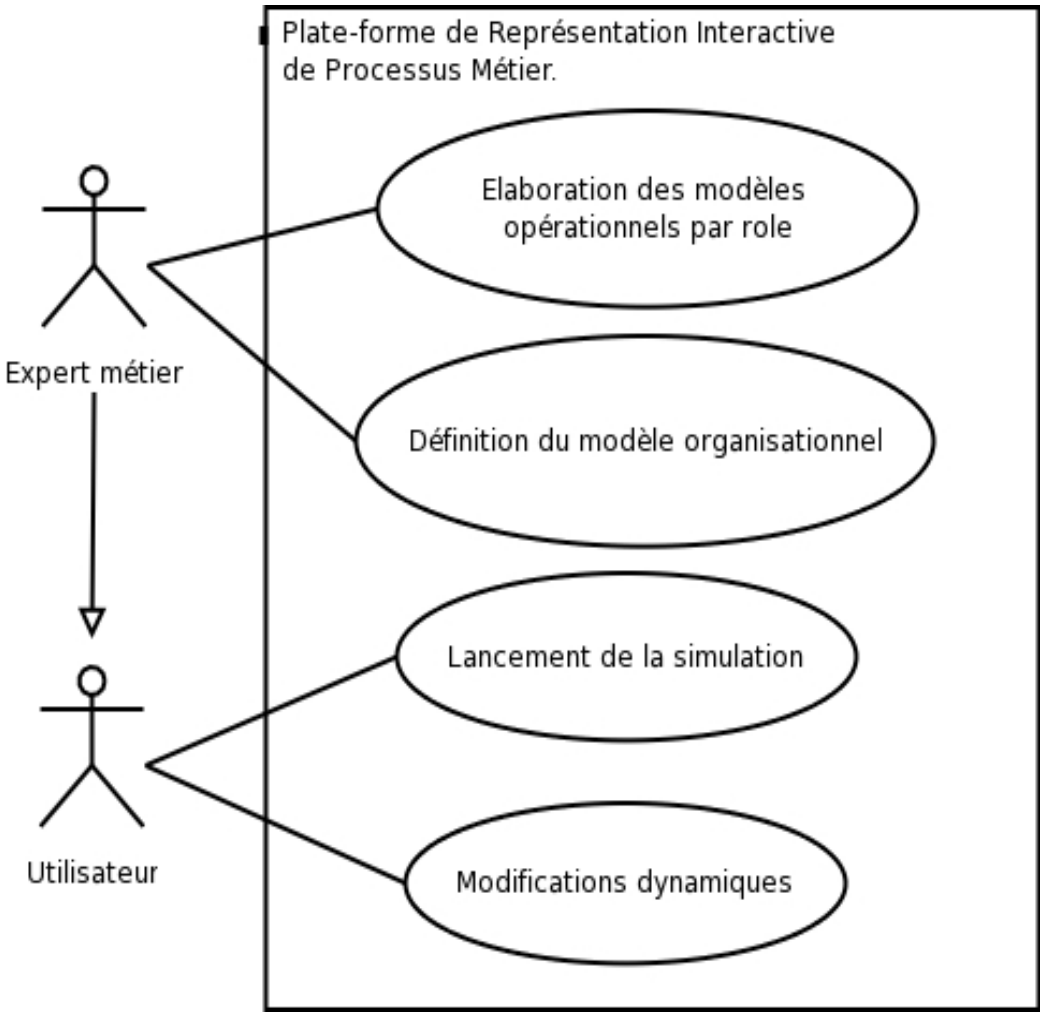


Figure IV.1 Diagramme des cas d'utilisation de la plate-forme de RIPM.

4.1.1. Le point de vue de l'expert métier.

L'expert métier est une personne qui a la maîtrise des processus métier de son entreprise. Dans les grandes structures, il appartiendra à des services nommés organisation ou maîtrise d'ouvrage. Ailleurs, ce sera une personne de la direction ou le chef d'entreprise lui-même.

La plate-forme devra, tout d'abord, lui permettre de créer les modèles opérationnels des différents rôles du ou des processus à représenter. Il utilisera le formalisme que nous avons défini au I.4.a avec les enrichissements vus au II.1 qui sont nécessaires. Puis, dans un second temps, elle lui permettra de définir un modèle organisationnel.

Ce peut être une définition précise de ce qui se passe en réalité dans une organisation comme dans le cas de son utilisation par un expert maison. Cette façon d'utiliser la plate-forme sera tester sur le cas d'application : "Représentation de processus métier d'une Agence Résidentiels de France Télécom". C'est pour ce type d'utilisation que nous avons noté une généralisation d'expert en utilisateur puisque dans ce cas il jouera les deux rôles.

Ou bien, une définition beaucoup plus générale d'un processus dit typique. Elle permettra de produire une représentation de base que l'utilisateur pourra raffiner selon ses désirs. Ce style d'utilisation de la plate-forme sera éprouvé dans l'application à la "Personnalisation de l'offre de services télécoms auprès de PME".

4.1.2. Le point de vue de l'utilisateur.

L'utilisateur pourra lancer la représentation avec les paramètres de visualisation par défaut ou définir ses propres paramètres. Une possibilité de définir un scénario particulier lui sera également proposé.

Un scénario correspond à la possibilité de faire varier des paramètres de la représentation interactive suivant un script qui est fonction du temps. L'utilisateur pourra visualiser, par exemple, le comportement global d'un processus suite à une augmentation du nombre d'instances pendant une semaine avec une absence de deux jours d'un acteur. Et ainsi voir le temps de retour à une situation normale.

L'utilisateur doit également pouvoir effectuer n'importe quelles modifications sur les modèles opérationnels et organisationnels pendant l'animation du ou des processus par la plate-forme. Il pourra donc, par des interfaces dédiées qui pourront apparaître en cliquant sur les objets de la représentation, faire des modifications comme, entre autres, reformuler les comportements des rôles, ajouter ou supprimer des acteurs, augmenter ou diminuer en nombre ou en volume les instances, changer la représentation graphique...

C'est sur ce dernier point que nous attendons beaucoup des possibilités offertes par une architecture basée sur les Adaptive Object-Models (AOM).

4.2. ARCHITECTURE DE LA PLATE-FORME.

L'architecture générale de la plate-forme est constituée d'un ensemble de composants logiciels qui permettent d'effectuer les principales fonctions que nous avons définies au niveau du comportement extérieur de ce système.

Cette architecture générale sous-entend que nous utiliserons le style architectural des AOM appliqué à des agents d'une plate-forme de Système Multi-Agents (SMA).

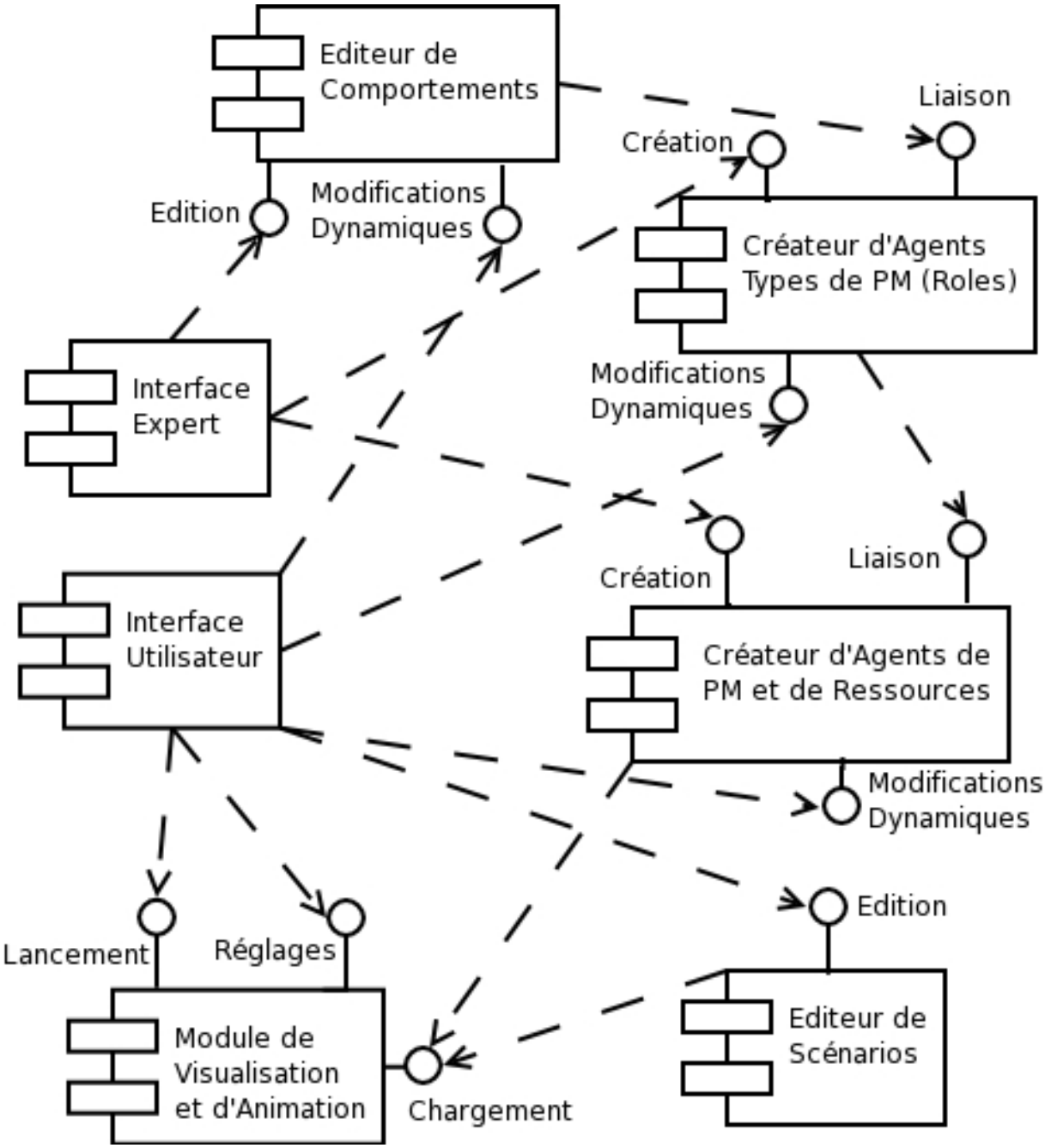


Figure IV.2 Diagramme des composants de la plate-forme de RIPM.

La figure IV.2 nous montre ces différents composants avec leurs interfaces. Nous y trouvons :

- un éditeur de comportements : il permet à l'expert de spécifier à l'aide des primitives de BPM les comportements souhaités pour les différents rôles du ou des processus à simuler. Cet éditeur permet de créer les agents porteurs de comportements (comme C_appro sur la figure III.10). On laisse une possibilité de modifications dynamiques par l'utilisateur s'il en a les capacités (ou si l'utilisateur est un expert).

- un générateur d'agents-types de PM : il va créer les agents-types de PM, c'est à dire les agents conservant la définition d'un rôle à l'image d'APPRO_TYPE dans notre exemple précédent. Leurs comportements sont externalisés dans les agents de comportements et on utilise un pointeur pour les reliés. Il est donc nécessaire de créer les comportements avant les agents-types. Il nous semble plus convenable que tous les agents de PM jouant le même rôle aient la même représentation. Cette dernière sera donc définie au niveau de l'agent-type. Une interface de modifications dynamiques par l'utilisateur est prévue. Il est donc possible, au cours de la représentation interactive, de modifier la représentation graphique des rôles ou de leur attribuer un autre comportement (par modification du pointeur vers un autre comportement).

- un générateur d'agents de PM et d'agents ressources : il s'agit ici de créer les agents simulant l'organisation réelle comme l'agent APPRO de la figure III.10. On va donc y définir en nombres et en qualités les agents de PM endossant un ou plusieurs rôles définis au niveau des agents_types. Les agents de ressources, quant à eux, simulent l'occupation de ressources organisationnelles. A leurs créations, il suffira de définir leurs représentations graphiques et de les attribuer à une unité organisationnelle ou de les dédier à un agent précis. Ce composant permettra également de disposer ces deux types d'agents sur l'espace de représentation. Là aussi, une modification dynamique de cette partie organisationnelle est prévue. On y donnera la possibilité d'ajouter, de modifier ou de supprimer des agents de PM et des agents de

ressources. Pour ces derniers, on pourra aussi modifier leurs affectations.

- un éditeur de scénario : il permettra à l'utilisateur de spécifier, sous la forme d'un script fonction du temps, des modifications tant opérationnelles qu'organisationnelles qu'il souhaite voir se succéder.

- un module de visualisation : il prend en charge la visualisation des agents de PM et des agents de ressources du ou des processus métier par le biais de l'utilisation d'une plate-forme de SMA. L'utilisateur en commandera le lancement et pourra apporter des réglages particuliers permettant d'adapter la simulation à ses besoins comme par exemple l'accélération ou le ralentissement du temps.

4.3. L'EDITEUR DE COMPORTEMENT.

C'est le composant principal pour l'expert métier puisqu'il va y élaborer les modèles opérationnels de chaque rôle.

4.3.1. Les agents 'Comportement'.

Même si l'expert pourra commencer par choisir le rôle qu'il veut définir, cet éditeur va créer un agent qui servira de support à ce comportement. On respecte ainsi l'architecture des AOM qui permettra de modifier dynamiquement le comportement d'un rôle.

Cet éditeur est basé sur le pattern 'Composite' (voir chapitre III.1.d). Il doit permettre de former des comportements sous forme de compositions à partir des primitives de processus métier.

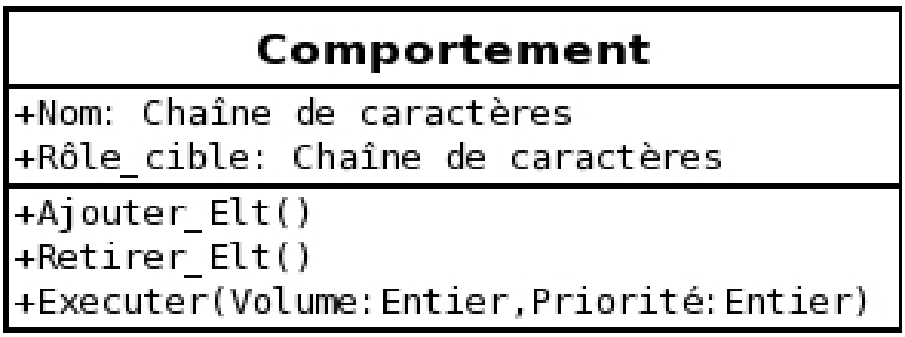


Figure IV.3 Diagramme de classe d'un comportement.

Les compositions et les primitives devront partager la même interface puisqu'on peut avoir, selon les cas, soit une composition soit une primitive seule. Du fait du souhait de gérer les notions de volumétrie et de priorité des instances d'exécution de processus, nous proposons d'utiliser une méthode du type Executer(Volume : entier, Priorite : entier).

Ces deux paramètres vont nous servir à différencier ces instances tant au niveau de leurs durées d'exécution qu'au niveau de leurs représentations graphiques. Ainsi, la volumétrie va agir sur la taille du motif représentant l'instance alors que la priorité pourra être déclinée suivant des codes couleurs (vert : basse , orange : moyenne, rouge : forte).

L'attribut 'Rôle_Cible' sert à donner le rôle qui pourra utiliser ce comportement et donc éviter des attributions inadaptées. Il permet également de préparer plusieurs types de comportements et de les substituer rapidement.

4.3.2. Les primitives de Processus Métiers.

Ces primitives sont des classes d'objets qui reprennent les éléments de base des modèles opérationnels (tâche, réception et émission de messages, test, synchronisation,...). Elles auront en attributs les données nécessaires à leurs personnalisations aux cas d'utilisations souhaités (la durée d'une tâche fixe, le rôle destinataire d'une émission de messages,...). Voyons les plus courantes :

- Les primitives de tâches : elles sont au nombre de deux, la tâche à durée fixe et celle à durée variable en fonction de la volumétrie de l'instance d'exécution à traiter.

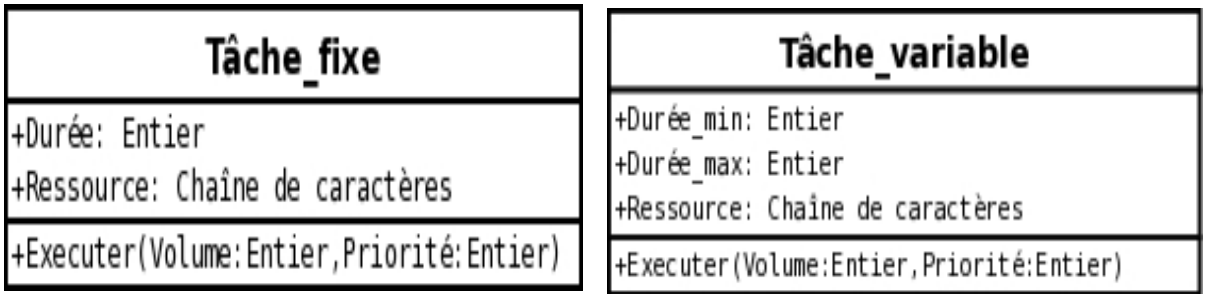


Figure IV.4 Diagrammes de classes des primitives de tâches.

Leur fonction est de simuler l'occupation d'un acteur effectuant une tâche donnée. Elles sont de durées fixes ou variables selon qu'elles sont dépendantes ou non de la volumétrie. Pour les tâches variables, on appliquera une fonction linéaire qui fera correspondre la volumétrie et la durée.

Si une ressource est spécifiée, l'agent devra tout d'abord en chercher une de disponible parmi celles qui sont mises à sa disposition (on utilisera un système de négociation entre

agents logiciels). S'il n'en trouve pas, cette instance d'exécution du processus reste bloquée en amont de la tâche jusqu'à la libération d'une d'entre elles. En attendant, l'agent pourra traiter une autre instance.

- Les primitives de communication : ce sont l'envoi et la réception de messages.

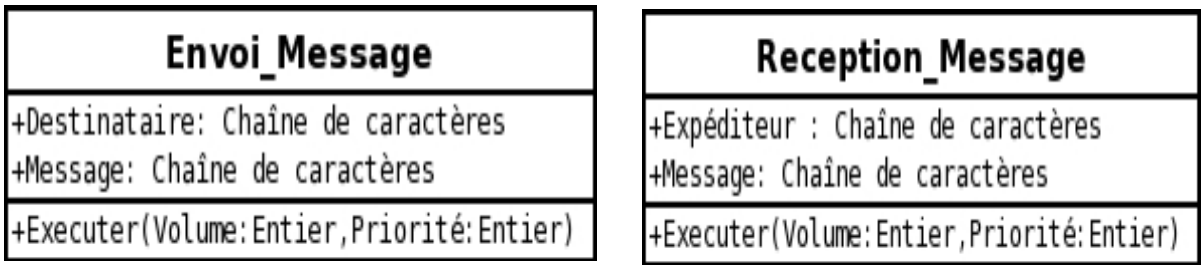


Figure IV.5 Diagrammes de classes des primitives de communication.

Les destinataires et les expéditeurs sont les rôles impliqués. L'attribut message contient un identifiant du type de message (commande, devis, autorisation,...). Il est utile pour distinguer les messages en cas de réceptions multiples d'un même rôle expéditeur à différent stade d'un même processus.

On utilisera pour les réaliser les moyens de communication de la plate-forme de SMA. Une communication par les rôles semblables à celle utilisée dans les modèles d'organisation AGR : Agent-Groupe-rôle [FG98] nous paraît bien convenir.

La réception d'un message est une primitive qui bloque l'instance d'exécution jusqu'à la survenue du message attendu. L'envoi lui n'est pas bloquant et introduit, s'il n'est pas la primitive terminale de son comportement, un parallélisme dans l'instance d'exécution.

- Les primitives de structures de contrôle : pour permettre de saisir tous les comportements, nous avons besoins de créer les principaux types de structures de contrôle qui sont le 'OU' exclusif et le 'ET' sous ses deux formes : le parallélisme et la synchronisation. La distinction sur la nature (données ou événements) des éléments testés faite par BPMN n'a pas

lieu d'être ici puisque nous n'effectuerons qu'une simulation de la décision d'ordre statistique.

Ces primitives vont agir sur les ramifications du graphe que forme un processus métier. L'attribut principal sera l'arité, c'est à dire le nombre de chemins en sortie ou en entrée dans le cas unique de la synchronisation.

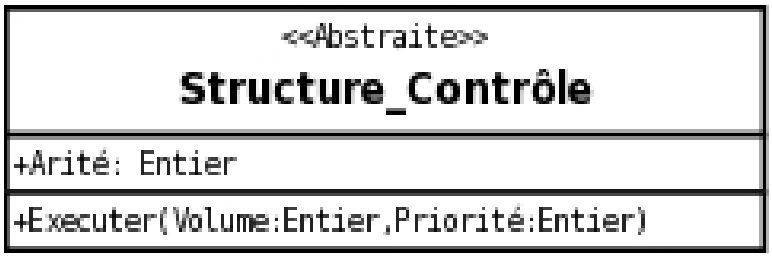


Figure IV.6 Diagramme de la classe abstraite des structures de contrôle.

En fonction de l'arité, il sera créé une structure variable qui conservera les pointeurs vers les successeurs. Pour les structures de type 'OU' exclusif on ajoutera la probabilité de survenance. C'est à partir de cette donnée qu'à chaque exécution un tirage aléatoire déterminera le chemin à prendre.

Pour les primitives correspondant au 'ET' de BPMN on s'aperçoit que nous aurons à créer deux classes dont les fonctions sont très différentes. Le parallélisme déclenchera le parcours de plusieurs chemins ensembles, l'arité la plus grande est sur les sorties. Alors que pour la synchronisation, qui matérialise un point d'attente de la fin de plusieurs branches, nous avons une grande arité en entrée.

Nous pensons qu'il n'y a pas lieu de créer une classe pour le 'OU' inclusif. Cela reviendrait à traiter trop de cas possibles (3 pour une arité de 2, 7 pour 3,...). Nous préférerons former cette primitive à partir de combinaisons de 'ET' de parallélisme et de 'OU' exclusif.

4.4. LES AGENTS-TYPES DE PROCESSUS METIER.

Ces agents servent à la définition complète des rôles. Ils sont porteurs de la relation vers le comportement ainsi que des représentations graphiques. Nous avons fait le choix de distingués les rôles internes et externes, il y aura donc une classe pour chacun.

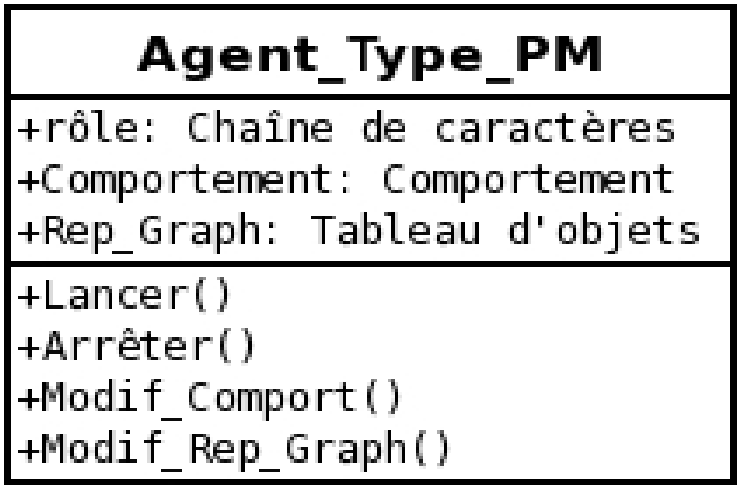


Figure IV.7 Diagramme de classe des Agents Types de PM (version interne).

La représentation graphique se doit d'être définie à ce niveau si l'on souhaite que les agents jouant le même rôle la partagent. Dans la version interne, nous avons pensé qu'il fallait prévoir une déclinaison d'un même motif qui variera en fonction de la charge de l'agent de PM. Il serait souhaitable de représenter au moins quatre états : surchargé, actif, au repos et désœuvré.

La version externe sera identique mais ne comportera qu'une seule représentation graphique qu'on veillera à être significative de cette distinction.

4.5. LES AGENTS INTERNES DE PROCESSUS METIER.

Ces agents vont récupérer le ou les comportements du ou des rôles qu'ils endossent et les exécuter dans leurs environnements propres.

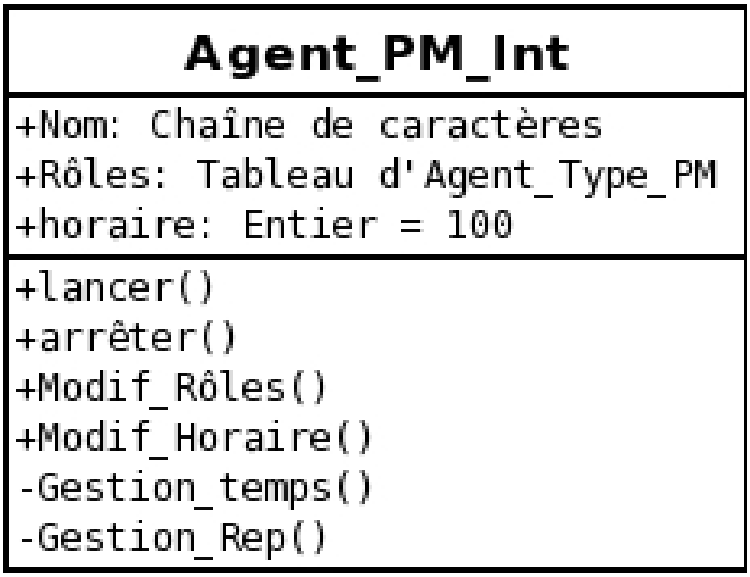


Figure IV.8 Diagramme de classe des Agents Internes de PM.

Ces agents seront du type réactif. En effet, c'est par la réception d'un message qu'ils vont exécuter le comportement attribué à leur rôle. Il faudra donc prévoir une boucle de perception qui scrute les messages en arrivée. Dans le cas d'acteurs à rôles multiples, cette boucle devra s'adapter à la prise en charge de messages distincts en conservant la répartition temporelle entre les rôles donnée par l'expert.

C'est le premier élément du comportement des rôles internes qui va donner les informations sur la nature du message déclenchant le comportement. Dans la majorité des cas, c'est un message provenant d'un rôle externe ou d'un autre rôle interne. S'il s'agit du déclenchement d'une règle métier, on fera générer le message par un agent spécialement conçu à cet effet de manière à conserver une classe unique d'agents internes de PM.

Il est nécessaire de simuler le temps de présence de l'acteur. Par défaut, il est à 100 qui correspond au temps plein. S'il est à moins on prévoira de désactiver l'agent en conséquence.

L'agent doit gérer sa représentation graphique lui-même. En fonction d'un indicateur

sur le taux d'occupation de l'agent sur une fenêtre de temps glissante, sa représentation se modifiera. Elle ira de 'désœuvré' s'il est très peu occupé à 'surchargé' si, au contraire, il l'est beaucoup. Si l'on n'atteint pas ces extrêmes, la représentation alternera entre actif et à au repos.

Il arrive que plusieurs instances d'exécution de processus puissent être traitées simultanément par le même agent. C'est le cas, par exemple, quand une instance est bloquée par l'attente d'un message et que l'agent en prend une autre en charge. Pour avoir un rendu représentatif de la réalité, il faut prévoir d'utiliser des mécanismes de synchronisation de l'exécution de ces instances.

4.6. LES AGENTS EXTERNES DE PROCESSUS METIER.

Ils différent des agents internes par le fait qu'un seul agent simule l'ensemble des acteurs réels ayant ce rôle. Donc ici aucun système de synchronisation n'est à prévoir puisqu'il n'y a pas de contrainte de ressource sur le temps.

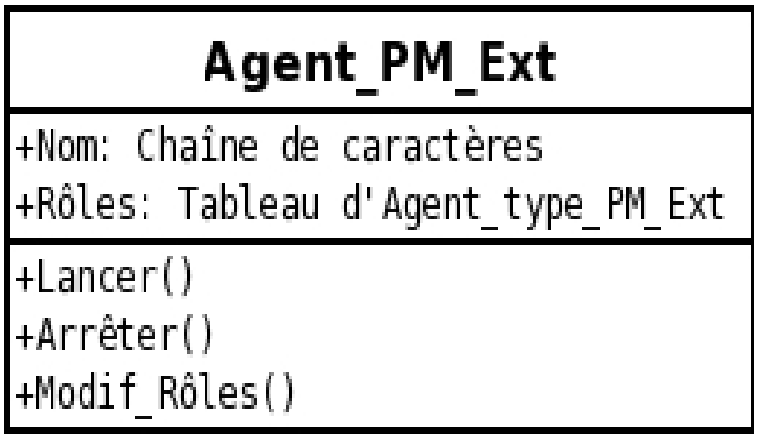


Figure IV.9 Diagramme de classe des Agents Externes de PM.

La représentation d'un agent externe de PM sera unique. Nous ne voyons pas d'états particuliers à distinguer.

Nous avons prévu la possibilité d'attribuer plusieurs rôles (on pourrait dire des sous-rôles). En effet, les rôles externes étant des rôles génériques (Banque, Clients,...) il est possible qu'ils interviennent à différents stades d'un même processus métier sans que cela ne forme un comportement continu. Prenons pour exemple le rôle du Client dans un processus de gestion des commandes. Il peut demander un devis, passer une commande ou l'annuler. Ces trois actions se doivent d'être décomposée en trois rôles distincts de manière à générer les trois types d'événements suivant les données statistiques de l'expert. Il peut également avoir à réagir à une sollicitation d'un rôle interne comme pour une validation. Dans ce cas il doit pouvoir scruter la réception du message pour lancer le comportement souhaité.

4.7. LES AGENTS RESSOURCES ORGANISATIONNELLES.

Ces agents ont pour but de simuler l'occupation des ressources organisationnelles (télécopieurs, ordinateurs, téléphones,...). Ils ont une notion d'appartenance qui permet de définir quels agents ont le droit de les utiliser.

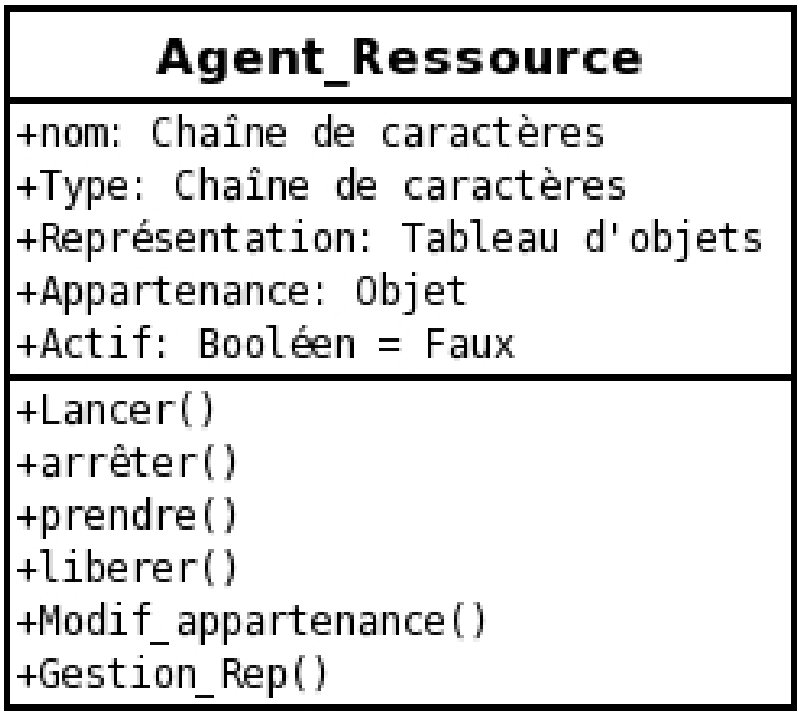


Figure IV.10 Diagramme de classe des Agents de Ressources.

Leur fonction principale est de garder l'état de la ressource qui est donné par l'attribut 'Actif' un booléen qui est mis à la valeur 'Faux' au lancement de l'agent. Les deux opérations 'Prendre' et 'Liberer' viennent le modifier. Cet attribut joue un rôle important lors de la recherche de ressources disponibles par les acteurs. C'est pour cette raison que nous l'avons mis en visibilité publique.

Le type correspond à la dénomination générique de la ressource (ordinateur, télécopieur,...) alors que le nom identifie précisément la ressource (fax du bureau B15, téléphone de Monsieur X,...). Il doit être identique à celui utiliser au niveau des primitives des tâches. Il servira dans le système de recherche de ressources libres.

Nous avons choisi de faire varier la représentation graphique de ces agents. Une

représentation lorsqu'ils sont actifs et une autre quand ils ne le sont pas.

En ce qui concerne l'attribut 'Appartenance' il y aura trois possibilités de valeurs :

- une référence à un Agent_PM_Int : quand il s'agit d'une ressource dédiée à un acteur précis.

- une référence à un Agent_Type_PM : quand la ressource est partagée par tous les acteurs d'un même rôle. Comme, par exemple, le fax du service commercial que se partagent tous les commerciaux.

- il restera vide si la ressource est partagée par toute l'organisation.

4.8. L'AGENT CONTROLEUR DE L'ANIMATION.

C'est à lui que nous allons déléguer les tâches de génération des événements déclencheurs et de gestion du temps simulé.

En effet, nous constatons que les modèles opérationnels individualisés des différents rôles débutent toujours par une réception de message d'un autre rôle. Même si nous avons inclus les rôles externes qui sont à l'origine du déclenchement des processus métiers internes, il nous faut gérer la génération des événements déclencheurs qui animeront les processus représentés.

Pour ce faire, nous proposons de créer un agent spécialement conçu pour envoyer des messages aux rôles contenant un début de processus. Il devra respecter la cadence et les paramètres de volume et de priorité donnés par l'expert et permettre de les modifier en cours de simulation par une interface qui sera visible sur l'écran de l'animation.

Ceci nous impose de créer une nouvelle primitive spécifique de début de processus qui est en faite une spécialisation de la primitive de réception de message. La différence c'est qu'elle recevra des messages de l'agent contrôleur lui donnant les informations sur les instances d'exécutions du processus à lancer.

Figure IV.11 Diagramme de classe de la primitive de début.

L'attribut 'Message' contiendra un identifiant du processus métier dont la primitive marque le début. A sa réception, la primitive 'Début' passera les paramètres à la primitive suivante dans le comportement via la méthode 'Executer'.

Nous pensons que cet agent peut aussi prendre à sa charge la gestion du temps dans la

simulation. En effet, par son aspect d'ordonnanceur des instances d'exécutions de processus, il est très impacter par toute modification du temps simulé. Il proposera donc une interface de réglage d'une échelle du temps de type curseur permettant de ralentir ou d'accélérer la simulation en cours. Il diffusera toute modification sur cette échelle par des messages à l'attention de tous les agents acteurs des processus (Agent_PM_Int et Agent_PM_Ext).

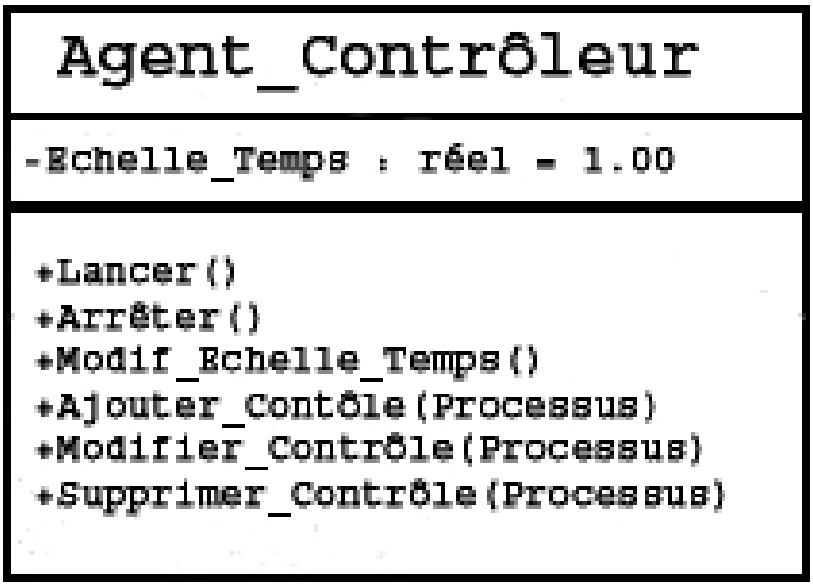


Figure IV.12 Diagramme de classe de l'Agent Contrôleur de l'animation.

L'attribut Echelle_Temps est un réel qui donne le rapport entre le temps réel et le simulé. Il est à 1 par défaut, c'est à dire que les simulations débuteront en temps réel. Il faut prévoir de faire partir l'échelle à 0 pour pouvoir arrêter l'animation si l'utilisateur le souhaite. Il est nécessaire de donner de nombreuses possibilités en matière d'accélération. L'utilisateur souhaitera vraisemblablement observer ses processus métier sur plusieurs heures ou jours dans un laps de temps de quelques minutes. Il est possible qu'il soit intéressant de ralentir l'animation le temps de voir ce qui se passe sur des parties de processus très rapides, nous ajouterons, à cet effet, quelques graduations telles que ¼, ½, ¾.

Cet agent bénéficiera des méthodes permettant de créer, de modifier et de supprimer son contrôle sur le déclenchement des processus à animer. Elles géreront les notions de cadence, de volumétrie et de priorité que nous avons détaillées au chapitre II.

4.9. LES AGENTS DE FILE D'ATTENTE EN RECEPTION.

La communication entre les agents par la notion de rôle atteint ses limites dès lors que plusieurs agents jouent le même rôle. Dans ce cas, tous les agents de même rôle reçoivent les mêmes messages risquant de provoquer des exécutions multiples d'une même instance d'exécution du processus.

Nous préconisons donc de créer un agent de réception pour chaque rôle interne (les rôles externes étant simulés par un seul agent) qui va prendre en charge par délégation la réception des messages adressés à ce rôle précis. Chaque Agent_PM_Int demandera à cet agent de lui délivrer un nouveau message chaque fois qu'il aura la possibilité d'en traiter un.

Cet agent sera aussi en charge de gérer la file d'attente de ces messages en fonction de leurs degrés de priorité. Il assurera la représentation graphique de cette file d'attente sur l'écran de l'animation a l'aide de motifs variant en fonction de la volumétrie globale des messages en attente.

Figure IV.13 Diagramme de classe des Agents de file d'attente.

Les messages reçus seront stockés dans une table et trier en fonction du paramètre de priorité. En cas d'égalité de priorité, on classera les messages du plus anciens au plus récent. La méthode 'Delivrer_message' renverra les paramètres du message en tête du classement à l'agent qui en fera la demande. Ce message sortira alors de la table et on fera avancer tous les messages d'une place. Cette technique évite d'utiliser une communication par message qui

viendrait perturber le bon fonctionnement de l'Agent_File_Attente.

Du point de vue implémentation, cet agent sera créé en même temps que le premier Agent_PM_Int. Il faut donc vérifier sa présence à chaque création d'un Agent_PM_Int, et si ce n'est pas le cas il faut le créer. Cette vérification doit être faite également lors de la modification des rôles sur les Agent_PM_Int.

CONCLUSION.

Au terme de ce mémoire, nous avons proposé une solution qui nous semble correspondre aux attentes du projet de "Représentation Interactive de Processus Métier". Cette part d'incertitude provient du fait d'avoir travaillé sur un projet de recherche qui ne s'est pas concrétisé.

Il aurait été plus prolifique de pouvoir confronter nos idées avec ses instigateurs. Nous aurions eu accès à leurs recherches précédentes dans le domaine des processus et de leurs modèles. Cela nous aurait permis de partir sur ces bases pour aller plus en profondeur et proposer une solution plus étoffée.

Néanmoins, la solution que nous avons proposée remplie les fonctions principales permettant l'animation et la modification dynamique de processus métier. Pour cela, nous avons défini une notation permettant de modéliser les processus métiers en incluant une vue organisationnelle qui s'est avérée nécessaire à une animation basée sur des agents logiciels. Nous avons décidé d'appliquer le concept des A.O.M. à notre architecture logicielle pour lui donner la flexibilité nécessaire aux modifications dynamiques. A ce stade, il faudrait faire l'implémentation d'un prototype pour valider notre solution sur les cas d'applications qui ont été définis.

Nous sommes convaincus que le mode de simulation proposé dans cette étude est très pertinent. Par son aspect graphique, il permet de se rendre compte rapidement et simplement du fonctionnement de processus métiers en situation d'exécution. Les larges possibilités de modifications en cours d'animation seront très apprécier des décideurs qui auront, dans l'instant, la vision des incidences sur l'ensemble.

Il est fort probable que des animations de ce type se rencontreront dans les produits de BPM dans les années à venir.

ANNEXE I. Contrat de recherche.

Contrat de recherche externalisée : FT/R&D –GREYC

(Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de l'Université de Caen)

Représentation interactive de processus métier

Annexe 1 : Annexe technique

Un processus métier est un ensemble d'activités qui sont mises en œuvre pour atteindre un objectif. Un processus métier peut être soit intra entreprise, soit entre l'entreprise et ses clients ou soit entre l'entreprise et ses partenaires.
Le déroulement séquentiel ou parallèle, synchronisé ou non des activités d'un processus peut être décrit à partir de diagrammes de processus métier de type BPMN (Business Processus Management Notation)
L'animation est un type de représentation de processus métier qui permet de montrer le comportement des processus lors de simulations. Elle met en jeu des instances des activités décrites dans les diagrammes de processus métier BPMN ou dans les diagrammes d'activité d'UML(Unified Modeling Language). Mais ces diagrammes ne décrivent pas le comportement des instances de processus qui se dérouleront en parallèle lors de simulations ou d'exécution réelle de ces processus.
Les expérimentations précédentes [1] [2] sur l'animation de processus métier qui ont été menées à FTR&D ont montré que nous pouvions tirer deux constats:

1) *il faut enrichir les notations existantes de type BPMN,*
2) *Il faut offrir un cadre au concepteur/utilisateur pour exploiter ces nouvelles notations.*

L'enrichissement des notations existantes de type BPMN est nécessaire à la fois pour produire des modèles exécutables, mais aussi pour décrire des représentations animées de ces modèles et enfin pour détailler les éléments nécessaires aux modifications interactives souhaitées par l'utilisateur.

La représentation de processus métier est interactive si on est capable de prendre en compte, de façon dynamique, des évolutions des processus métier initialement représentés.
Cette interactivité pourra être rendue par un travail sur des questionnaires intelligents et par l'étude d'une architecture logicielle suffisamment modulaire et flexible sur laquelle pourront s'appuyer ces questionnaires intelligents.

Le travail de recherche proposé consiste donc

- *(i) à enrichir les notations existantes de type BPMN,*
- *(ii) à étudier les moyens à mettre en œuvre permettant de prendre en compte une représentation interactive de processus métier : questionnaires intelligents et architecture logicielle flexible.*
- *(iii) il devrait mettre en œuvre les résultats obtenus sur deux cas d'application : la promotion de services télécoms auprès des PME et l'amélioration des processus d'une Agence. Ces deux cas d'application utiliseraient des questionnaires intelligents permettant rapidement de prendre en compte l'existant des PME d'une part et le substrat des évolutions des processus de l'Agence d'autre part.*

La mise en œuvre des résultats obtenus sur les deux cas d'application implique la fourniture d'objets graphiques et sonores permettant de fabriquer des animations.
Les animations de processus proposées par les fournisseurs d'outils de BPM (Business Process Management) consistent le plus souvent à afficher des parties de diagrammes de processus métiers de type BPMN et à faire changer de couleur les éléments actifs de ces diagrammes.
Il semble que nous pouvons tirer beaucoup plus partie des technologies d'animations graphiques et sonores actuelles.
A partir d'un état de l'art dans le domaine de l'animation graphique et sonore, il sera intéressant de constituer un corpus d'éléments potentiellement intéressants pour les besoins du BPM, par le biais d'un marché avec une Société de Services en Infographie.

1- Introduction

Le management de processus métier ou en anglais BPM (Business Process Management) s'intéresse aux processus métier des entreprises et à toutes les solutions qui permettraient aux entreprises de dynamiser leurs ressources en accélérant la conception, la reconfiguration et l'exécution de leurs processus métier.

Les utilisateurs du BPM souhaitent avoir une vue de bout en bout de leurs processus en cours d'exécution, souhaitent mieux maîtriser l'évolution de leurs processus métier et pouvoir analyser l'impact d'une modification à un endroit sur l'ensemble de leurs processus. Ils souhaitent aussi pouvoir dimensionner leurs processus, observer le comportement réel de certaines files d'attente ou l'occupation de certains acteurs dans certaines configurations, pouvoir déceler rapidement la cause d'un problème, déterminer des indicateurs pertinents permettant le contrôle de leurs processus.

Les diagrammes de processus métier de type BPMN (Business Processus Management Notation) permettent de représenter un certain niveau de comportement des processus métier des entreprises. Ces notations sont compréhensibles par les analystes qui créent les processus, par les développeurs qui implémentent les technologies qui les mettront en œuvre et par les responsables qui auront à les piloter. Mais ces diagrammes ne décrivent pas le comportement des instances de processus qui se dérouleront en parallèle lors de simulations ou d'exécution réelle de ces processus. Des expérimentations précédentes [1] [2] qui ont été menées à FTR&D ont montré qu'il fallait par exemple annoter les diagrammes d'activités UML pour produire des modèles exécutables.

Un travail sur les notations diagrammatiques permettant de produire des modèles exécutables, de rendre compte des animations et de détailler les éléments nécessaires à des modifications interactives sera entrepris pour construire des modèles indépendants des outils et des plateformes.

Ces représentations de processus métier, complémentaires des notations existantes, devraient contribuer à une meilleure compréhension et une meilleure maîtrise des organisations et des systèmes complexes qui émergeront de l'assemblage d'éléments relativement simples au départ.

La représentation de processus métier devient à notre sens "interactive", si elle est capable de prendre en compte dynamiquement des évolutions non triviales des processus initialement représentés. Une étude sur une architecture logicielle permettant la prise en compte dynamique des évolutions de processus métier sera entreprise en liaison avec l'élaboration de questionnaires intelligents susceptibles de saisir l'essentiel des évolutions souhaitées.

Deux cas d'applications intéressant directement France Télécom serviront de support pour illustrer les résultats de cette collaboration : la personnalisation de l'offre de services télécoms auprès de PME et l'animation de processus métier d'une Agence Résidentiels France Télécom.

Les animations de processus proposées par les fournisseurs d'outils de BPM consistent le plus souvent à afficher des parties de diagrammes de processus métiers de type BPMN et à faire changer de couleur les éléments actifs de ces diagrammes. L'utilisateur a une vue d'ensemble grossière de ce qui se passe qui est de l'ordre du "sablier qui apparaît quand une application informatique s'exécute".

Il semble raisonnable de penser que nous pourrions tirer beaucoup plus partie des technologies actuelles dans le domaine de l'animation graphique et sonore pour mieux couvrir les besoins des différents acteurs du BPM.

Un marché avec une Société de Services en Infographie nous permettra d'obtenir les éléments graphiques et sonores utiles pour cette illustration.

2- Problématique

Celle-ci s'articule autour de deux axes principaux :

- l'enrichissement des notations existantes pour le BPM,
- l'élaboration de questionnaires intelligents et l'étude d'une architecture logicielle permettant la prise en compte dynamique des évolutions de processus métier.

2-1 Enrichissement des notations existantes pour le BPM

Une étude des standards existants en matière de BPM et autres systèmes de notation similaires (par exemple les diagrammes d'activité UML) sera effectuée. Celle-ci devra mettre en évidence et proposer les compléments nécessaires pour les besoins du projet.

Il faudra en particulier :

- pouvoir spécifier les processus que l'on souhaite simuler (incluant les processus externes à l'entreprise indispensables à la simulation) et l'ensemble des éléments que l'on souhaite représenter,
- pouvoir spécifier les ressources organisationnelles, humaines, matérielles et informationnelles qui interviennent dans les processus que l'on souhaite simuler.
- pouvoir prendre en compte les concepts de scénario que l'on souhaite voir se dérouler,
- pouvoir spécifier les concepts "d'intentions et stratégies métier" afin de pouvoir prendre en compte leur influence sur les changements des processus métiers modélisés. [3]
- pouvoir spécifier le cadre espace - temps nécessaire à l'exécution des processus. Une des limites de l'espace des démonstrations sera la taille de l'écran. Une autre pour le temps sera la durée de la démonstration envisagée. Les informations sur les durées des processus permettent de produire une animation vraisemblable dans le temps relativement court de la démonstration mais elles vont dépendre de l'objectif principal de la démonstration. Pour des démonstrations "didactiques" il faudra accélérer les processus trop lents et ralentir ceux qui sont trop rapides pour être perçus. Les démonstrations "de performance" où l'objectif est d'observer le comportement réel de certaines files d'attente ou l'occupation de certains acteurs dans certaines configurations de l'entreprise, des mises à l'échelle seront indispensables.
- pouvoir spécifier les stratégies de fonctionnement (priorité d'exécution des processus) et les pratiques d'intervention de l'entreprise (en cas de concurrence entre acteurs pour le traitement d'un processus).

2-2 Elaboration de questionnaires intelligents et étude d'une architecture logicielle permettant la prise en compte dynamique des évolutions de processus métier.

L'élaboration de questionnaires intelligents permettant de saisir l'essentiel des évolutions souhaitées a pour objectif principal la rapidité de la saisie. La prise d'informations relatives au métier du prospect (pour le cas "personnalisation de l'offre de services télécoms auprès de PME") doit pouvoir se faire dans une durée comprise entre 10 et 60 minutes (10 minutes pour convaincre un client PME pressé).
La prise d'information permettant de saisir le substrat des évolutions des processus métier (pour le cas "représentation de processus métier d'une Agence Résidentiels France Télécom"**)** doit pouvoir se faire dans une durée raisonnable de plusieurs dizaines de minutes.
Ces prises d'information demandent en outre une bonne connaissance de l'expertise des commerciaux FT ou des processus métier d'une Agence, notamment aux fins de validation.

Les questionnaires intelligents travailleront par une interaction et un enrichissement des données basés sur l'exécution d'un scénario typique de processus par défaut.
Ils s'appuieront pour cela sur un modèle organisationnel de Système Multi-Agents qui pourra facilement représenter la coopération entre éléments de processus.
L'architecture sera rendue aussi modulaire que possible au moyen de principes issus de la Programmation Orientée Aspect.
Les bénéfices attendus sont surtout la possibilité de reconfigurer dynamiquement des descriptions de processus, et de leurs représentations.
La flexibilité du modèle d'exécution permettra de modifier de manière interactive la représentation en cours :

➔ en phase de réalisation de cette représentation afin de pouvoir tester plusieurs principes différents de représentation (par exemple pouvoir comparer des animations visuelles continues de comportements dynamiques de processus avec des animations semi-discrètes de type « dessin animé »)

➔ en phase d'exploitation de cette représentation pour prendre en compte des affinements et modifications des descriptions d'un processus métier par les utilisateurs.

3- Les cas d'application

3-1 La personnalisation de l'offre de services télécoms auprès de PME.

Pouvoir rapidement montrer à un client comment ses processus métiers pourraient se comporter s'ils intégraient des offres de services télécoms comme un site Internet pour la vente en ligne, l'utilisation d'un centre d'appel pour la prise de commande ou plus généralement l'utilisation d'un système de management de la relation avec la clientèle serait un moyen de marketing efficace pour promouvoir des offres de service télécoms auprès de PME.

L'éventail "PME" étant très riche, ce cas d'application nécessitera l'élaboration de questionnaires "intelligents" permettant de saisir le plus rapidement possible, en boucle de raffinement successif, l'essentiel de l'existant des processus métiers de la PME cliente.

3-2 Représentation de processus métier d'une Agence Résidentiels France Télécom.

L'intérêt de ce cas d'application est multiple. Il permettra d'avoir une vue de bout en bout des processus en cours d'exécution, d'observer le comportement "réel" de certaines files d'attente ou l'occupation de certains acteurs dans certaines configurations de l'Agence en vue d'un meilleur dimensionnement des processus. Il permettra aussi de mieux maîtriser l'évolution des processus et de pouvoir analyser l'impact d'une modification à un endroit sur l'ensemble des processus. Il permettra enfin de rechercher des indicateurs pertinents permettant de surveiller et contrôler un ensemble de processus afin de pouvoir réagir rapidement en cas d'incident.

Dans ce second cas d'application on retrouve la nécessité d'élaborer des questionnaires "intelligents" permettant de saisir le substrat des évolutions des processus métier (nouveaux contrats de service à respecter, suppression ou modification d'anciens processus, introduction de nouveaux processus) en vue de leur prise en compte rapide.

4- Références

[1] Mariano Belaunde, Maria-José Presso. "Méta-modèle d'un modèle de processus" Edition 2.0, T&I/FTR&D/DTL/TAL/03.109/MB

[2] Maria-José Presso, Roland Vivès ." Principes de passage d'un modèle vers une maquette". Livrable 5.4 V1 — décembre 2003, 85 pages.

[3] Objectifs d'Entreprise et Systèmes d'Information : Réflexion Conceptuelle et Cadre Méthodologique pour l'Analyse du Changement . Rapport final du CRE Université Paris I – Sorbonne, août 2002, 130 pages.

ANNEXE II – Présentation du GREYC.

Le Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (UMR 6072) est un laboratoire du département STIC du CNRS. Il regroupe environs 140 chercheurs de l'Université de Caen et de l'Ensi-Caen.

Ils sont regroupés en 7 équipes :

- Algorithmique
- Automatique
- DoDoLa (Données, Document et Langue)
- Image
- Instrumentation
- ISLanD (Interaction sémiotique : Langues, Diagrammes)
- MAD (Modèles, Agents et Décisions)

C'est au sein de l'équipe ISLanD que s'est déroulé le travail présenté dans ce mémoire. Ses sujets d'études sont les langues et les diagrammes vus comme les principaux vecteurs des interactions entre les hommes et les machines. Elle est dirigée par Anne Nicolle et s'articule autour de 3 thèmes :

- Syntaxe et rhétorique : ce groupe conçoit et produit des analyseurs s'appliquant aux mots comme aux phrases et aux différentes formes de documents. Son originalité est de ne pas utiliser de dictionnaires de mots lexicaux.

- Diagrammes, information et communication : ce thème étudie les notations diagrammatiques comme support de travail coopératif qu'il soit de nature didactique ou à vocation de modélisation(UML par exemple). Par leur étude sur le plan sémiotique, des propriétés permettant d'améliorer les interfaces homme-machine devraient se révéler.

- Modélisation du langage et de l'activité langagière : il s'agit ici de concevoir une plate-forme homogène d'outils d'étude linguistique permettant la modélisation du langage et des activités langagières.

BIBLIOGRAPHIE.

[Bec96] **BECK K.** - *Smalltalk Best Practice Patterns* - PrenticeHall, 1996.

[Coa92] **COAD P.** - *Object-oriented Patterns* - Communications of the ACM, 35(9), pp. 152-159, 1992

[Cru03] **CRUSSON T.** - *Business Process Management, de la modélisation à l'exécution - Positionnement par rapport aux architectures orientées services* - Intalio, 2003, http://www.application-servers.com/articles/bpm/livreblanc/IntalioBPM.pdf

[FG98] **FERBER J. et GUTKNECHT O.** - *A meta-model for analysis and design of multi-agent systems* - Proc. International Conference on Multi-Agent Systems, 1998

[Fow97] **FOWLER M.** - *Analysis Patterns: Reusable Object Models* - Reading, MA, Addison-Wesley, 1997

[FY98] **FOOTE B., YODER J.** - *Metadata and Active Object Models* - Proceedings of Plop98. Technical Report #wucs-98-25, Dept. of Computer Science, Washington University Department of Computer Science, October 1998

[GC02] **GLASSEY O., CHAPPELET JL, -** *Comparaison de trois techniques de modélisation de processus : ADONIS, OSSAD et UML* – Working paper de l'IDHEAP, http://www.idheap.ch/, 2002

[GHJV95] **GAMMA E., HELM R., JOHNSON R., VLISSIDES J.** - *Design Patterns: Elements of Reusable Object-Oriented Software* - Reading, MA, Addison-Wesley, 1995

[Joh98] **JOHNSON R.,** - *Dynamic Object Model* – (document non publié), http://st-www.cs.uiuc.edu/users/johnson/DOM.html, 1998

[JW98] **JOHNSON R., WOLF B.,** - *Type Object* – Pattern Languages of Program Design 3, Reading, MA, Addison-Wesley, 1998

[Ket98] **KETTANI N., MIGNET D., PARE P., ROSENTHAL-SABROUX C.** -*De Merise à UML* - Paris : Eyrolles, 1998

[MO95] **MARTIN J., ODELL J.** - *Object Oriented Methods: A Foundation* - Englewood Cliffs, NJ, Prentice Hall, 1995

[Raz01] **RAZARI R.** - *Outils pour les Langages d'Experts : Adaptation, Refactoring et Réflexivité* - Thèse de doctorat, LIP6- OASIS, Université Pierre et Marie Curie (Paris 6), Paris : 2001

[YBJ01] **YODER J., BALAGUER F., JOHNSON R.** - *The Architectural Style of Adaptive Object-Models* - ECOOP 2001 Workshop on Adaptive Object-Models and Metamodeling Techniques.