Thèse en Cotutelle

Entre :

Université Mentouri de Constantine
Faculté des sciences de l'ingénieur
Département d'Informatique

Université Pierre et Marie Curie de Paris 6
École doctorale : Informatique, télécommunication et électronique

Spécialité

Informatique

Présentée par

M. **Djamel Benmerzoug**

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE
ET
DOCTEUR de L'UNIVERSITE DE CONSTANTINE

Sujet de la thèse :

Modèles et outils formels pour l'intégration d'applications d'entreprises

Soutenue le 13 Décembre 2009 à l'université de Constantine

devant le jury composé de :

M. Fabrice Kordon (Pr.)	Univ. Pierre et Marie Curie de Paris 6	Directeur de thèse
M. Mahmoud Boufaida (Pr.)	Univ. Mentouri de Constantine	Directeur de thèse
M. Lionel Seinturier (Pr.)	Univ. Lille 1, Lille	Rapporteur
M. Mohamed Ahmed-Nacer (Pr.)	Univ. USTHB, Alger	Rapporteur
Mme. Béatrice Berard (Pr.)	Univ. Pierre et Marie Curie de Paris 6	Examinateur
M. Mohamed Benmohammed (Pr.)	Univ. Mentouri de Constantine	Examinateur

À mes parents,
ma femme et
mon petit fils Badreddine

Remerciements

Je tiens à remercier, tout d'abord, mes directeurs de thèse :

Monsieur **Mahmoud Boufaida**, Professeur à l'Université Mentouri de Constantine, qui a dirigé cette thèse, qui m'a amicalement conseillé tout au long de ce travail et qui m'a proposé un cadre de travail très favorable. Je le remercie d'avoir consacré beaucoup de son temps pour les nombreuses relectures de mon document et les articles publiés. Ses remarques m'ont permis de faire progresser ce travail.

Monsieur **Fabrice Kordon**, Professeur à l'Université Pierre et Marie Curie de Paris 6, pour l'encadrement de mon travail et son apport tant au niveau des connaissances qu'au niveau humain. Et aussi pour son encouragement, ainsi que son soutien tout au long de la thèse. Je le remercie également de m'avoir accueilli dans le laboratoire Lip6 (équipe MoVe) où nous avions eu de nombreuses discussions très fructueuses.

Je tiens également à remercier les membres du jury qui m'ont fait l'honneur de bien vouloir évaluer mon travail, et plus précisément :

Monsieur **Mohamed Benmohammed**, Professeur à l'Université Mentouri de Constantine, pour l'honneur qu'il m'a fait, en acceptant la présidence de ce jury.

Madame **Béatrice Bérard**, Professeur à l'Université Pierre et Marie Curie de Paris 6, pour avoir accepté de juger le présent document. Je la remercie également pour ses commentaires pertinents qui ont permis d'améliorer la qualité de mon document

Monsieur **Lionel Seinturier**, Professeur à l'Université de Lille 1, d'avoir accordé de son temps précieux pour se pencher sur mon humble travail et apporté des suggestions constructives pour l'amélioration de ce document.

Monsieur **Mohamed Ahmed-Nacer**, Professeur à l'USTHB Alger, d'avoir bien voulu se pencher sur ce travail et faire partie de ce jury. Je le remercie également pour le temps qu'il m'a accordé et pour ses commentaires constructifs qui ont permis d'améliorer la qualité de mon document.

J'associe mes remerciements à tous mes amis de l'équipe SIBC du laboratoire LIRE, pour leur sympathie, leur appui amical et pour leur aide spontanée.

Je remercie les membres de l'équipe MoVe du laboratoire Lip6 pour l'accueil chaleureux qu'ils m'ont toujours réservé.

Je remercie les gouvernements algérien et français pour leur support financier accordé à ce travail dans le cadre de l'accord-programme enregistré sous le numéro 05MDU640 et intitulé : "Conception de systèmes pour l'intégration des applications d'entreprise : Application aux systèmes d'information basés WEB".

Je remercie l'Agence Universitaire de la Francophonie (AUF), et en particulier le Bureau Europe de l'Ouest et Maghreb, pour m'avoir accordé une bourse pendant l'année universitaire 2007/2008, ce qui m'a permis de réaliser ce travail.

Enfin, je remercie les membres de ma famille et mes amis pour leur soutien et leur compréhension.

Paris, le 29 Octobre 2009.

Table des matières

Liste des figures

Introduction générale

1 Contexte de la thèse

L'évolution rapide des performances des réseaux d'ordinateurs, comme Internet et les Intranets de sociétés, est due aux besoins de plus en plus croissants du partage d'informations et de ressources à l'intérieur (intra-entreprise) et entre les différentes entreprises (inter-entreprise). En effet, les échanges entre les différentes applications d'entreprises sont devenus des enjeux majeurs pour les organisations qui doivent multiplier leurs canaux de communication (filiales, usines, partenaires, clients, fournisseurs, etc.). Les entreprises doivent à la fois connecter les nombreuses applications hétérogènes existantes, exploiter des données issues de systèmes d'information différents et définir de nouveaux processus tout en garantissant à terme la cohérence du système d'information [141].

A ce stade, les outils de l'Intégration d'Applications d'Entreprises (IAE [1]) conjointement aux Architectures Orientées Services (AOS [2]), fournissent une nouvelle manière de développer des applications conformes aux exigences d'Internet [82]. Cette utilisation permet alors de conserver les systèmes existants [3] et d'exposer ces programmes à travers Internet par de nouvelles technologies favorisant l'interopérabilité.

Intégration d'applications et processus métiers

L'IAE est un ensemble de solutions, de processus et de méthodes destinés à assurer l'intégration d'applications hétérogènes. L'objectif est d'augmenter l'adaptabilité du système d'information et de réduire les coûts de maintenance et de développement des interfaces [120]. La difficulté majeure relève du fait que ces applications ont été généralement conçues de façon indépendante.

La plupart des travaux tirés de la littérature sur l'intégration d'applications permettent de mettre en évidence quatre approches basées sur : 1) l'intégration des applications par les données, 2) les traitements, 3)les présentations et 4) les processus.

1. Plus connue sous le nom en anglais EAI pour *Enterprise Application Integration*
2. Plus connues sous le nom en anglais SOA pour *Service Oriented Architecture*
3. Plus connus sous le nom en anglais *legacy systems*

Les trois premières approches peuvent être considérées comme le résultat logique de la structuration des applications en couches, tandis que la quatrième approche, qui se décline par la suite en une combinaison des trois approches de bases, résulte de la mise en oeuvre d'un médiateur (appelé moteur de workflow), et qui permet ainsi d'interconnecter des applications via l'orchestration des processus.

L'intégration d'applications par les processus[4] recouvre la définition et la gestion des échanges d'information d'une entreprise à travers une sémantique des processus. Cette intégration définit comment est gérée une séquence d'événements. Ces derniers représentent des activités plus ou moins longues devant être complétées pour traiter une tâche faisant partie d'un processus métier. Afin d'implémenter ces activités, l'intégration des processus consisterait idéalement à pouvoir définir un modèle graphique du processus métier et à générer automatiquement l'intégration entre modèles, systèmes informatiques et acteurs [48] [17].

Les architectures orientées services

L'AOS est une architecture logicielle s'appuyant sur un ensemble de services simples. Son objectif est de décomposer une fonctionnalité en un ensemble de fonctions basiques (les services) fournies par des composants et de décrire le schéma d'interaction entre ces services [82].

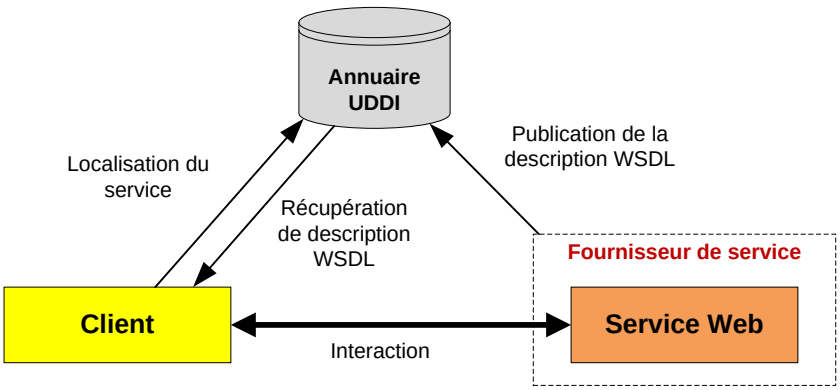

FIGURE 1 – *Déploiement, recherche et invocation des services Web*

Comme le montre la figure 1, la communication entre le client et le service passe par une phase de découverte et de localisation du service, à l'aide du protocole et d'annuaires UDDI [124]. Ces annuaires contiennent un ensemble de descriptions de services Web, utilisant le langage WSDL[34], basé sur XML. Le client interroge l'annuaire à l'aide de mots clés pour obtenir un ensemble de descriptions WSDL. Ces descriptions contiennent toutes les informations nécessaires à l'invocation du ser-

4. Plus connue sous le nom en anglais BPI pour *Business Process Integration*

vice (URL de localisation, description des fonctions et des types de données).

Un des défis des AOS est l'intégration de services pour la création de nouveaux services personnalisés [105].

En effet, si une entreprise ou un client requiert des fonctionnalités, et qu'aucun service n'est apte à les fournir, il devrait être possible de combiner ou de composer des services existants afin de répondre aux besoins de cette application ou de ce client [83][18]. C'est ce que l'on appelle la composition de services [3].

Les services Web semblent donc être bien adaptés aux problèmes sous-tendus par l'intégration des processus métiers. Ils permettent en effet, une intégration centrée sur les services, où les fonctions automatiques sont mises à la disposition de tous les utilisateurs éventuels, quelles que soient la technologie et les spécifications d'interfaces qu'ils utilisent [3].

Toutefois, ces approches présentent plusieurs limites dont voici les plus récurrentes [60][26] :

- Les services web sont passifs jusqu'à ce qu'ils soient invoqués ;
- Un service web a seulement connaissance de lui-même, mais pas celle de ses applications ou ses utilisateurs clients ;
- Un service web n'est pas adaptable, et il n'est pas capable de bénéficier des nouvelles capacités de l'environnement afin d'apporter des services améliorés.

L'autonomie, l'adaptation et la coopération, points faibles des services Web, sont par ailleurs des mécanismes qui ont largement été explorés dans le domaine des systèmes multi-agents. Plusieurs études décrivent à ce propos la pertinence de la combinaison des technologies agent et services Web [26][21][89].

Dans ce qui suit, nous faisons une brève présentation des systèmes multi-agents et de l'interaction dans ces systèmes. Nous rapprochons l'intégration d'applications de l'interaction multi-agents avant d'esquisser notre proposition pour l'intégration des processus métiers.

Les systèmes multi-agents

Les systèmes multi-agents (SMA) font partie de ce qu'il est convenu d'appeler "l'Intelligence Artificielle Distribuée" (IAD), branche relativement récente de l'IA. A la différence de l'IA classique qui modélise le comportement intelligent d'un seul agent (aspect individuel), l'IAD s'intéresse à des comportements intelligents qui sont le produit de l'activité coopérative de plusieurs agents (aspect collectif).

M. Luck et al. définissent un SMA comme étant "un réseau d'entités fortement couplé qui interagissent pour résoudre des problèmes dépas-

sant les capacités individuelles et la connaissance de chaque entité. Ces entités, souvent appelées agents, sont autonomes et peuvent être hétérogènes" [81].

L'interaction est une caractéristique très importante dans les SMA. La résolution collective d'un problème est le résultat de l'interaction coopérative entre les différents agents.

Elle est définie comme une séquence de messages cohérents et interdépendants entre eux, échangés dans le but de réaliser une tâche collective [59]. Chaque interaction est régie par un **protocole**, qui regroupe un ensemble de contraintes sur les échanges de messages, et dispose d'une sémantique propre [119].

Un **protocole d'interaction** est un ensemble de règles et de contraintes communicationnelles, associées à un ensemble fini de rôles qui seront joués par des agents. Il permet aux agents d'avoir des conversations, sous forme d'échanges structurés de messages [61]. Il décompose une tâche en sous-tâches et les distribue, suivant une stratégie donnée, aux agents du système.

Par ailleurs, l'intégration d'applications permet de gérer et d'automatiser des processus métiers entre plusieurs organisations. Dans ce contexte, par nature réparti et hétérogène, plusieurs organisations partenaires mettent en commun leurs processus et les font interagir selon certains **protocoles** pour rendre un nouveau service constituant une valeur ajoutée pour chaque partenaire. Ces protocoles correspondent par exemple à des protocoles de recherche de partenaires, des protocoles de négociation ou encore à des protocoles d'élaboration et d'exécution de contrat.

En effet, les protocoles d'interaction dans les SMA s'adaptent bien à la modélisation des problématiques (de répartition, d'ouverture, de flexibilité et de coordination des processus, ect.) sous-tendues par l'intégration d'applications.

Dans ce type d'applications, les protocoles sont donc identifiables et récurrents. Il est alors utile de les isoler afin de bien les étudier, les modéliser et les implémenter comme des entités à part entière pour permettre leur partage, leur recherche et leur invocation.

C'est la raison pour laquelle notre étude va s'intéresser aux solutions basées sur les SMA pour la modélisation des problématiques d'intégration d'applications. Dans cette thèse, nous nous intéressons à l'intégration d'applications par les processus. Cette intégration consiste à coordonner des processus métiers (issus de différentes organisations) afin d'atteindre un objectif commun. Notre objectif est alors de spécifier des propriétés et des protocoles d'interaction dédiés aux SMA, vérifier leurs comportements, et de les adapter dans le contexte de l'IAE.

2 Problématique

Le problème principal qui se pose pour l'IAE est de ne pas faire intégrer seulement des applications développées avec des technologies nouvelles. Il faut également faire coopérer des systèmes déjà existants et qui sont toujours opérationnels. Le but de notre travail est donc de prévoir une couche au dessus de ces systèmes afin qu'ils puissent coopérer de façon efficace.

Pour faire face à ces problèmes, les services Web fournissent une solution prometteuse. Ils consistent à exposer sur le réseau d'Internet, une ou plusieurs applications. Ces services peuvent proposer des fonctions très simples (du type requête/réponse) ou un ensemble complet d'outils, permettant d'aller jusqu'à la composition des services pour proposer une application complète.

Comme nous l'avons évoqué dans la section précédente (et nous le détaillerons dans le chapitre suivant), les services Web présentent plusieurs limites, surtout dans le cas d'intégration d'applications d'entreprises, qui nécessite l'exécution d'une interaction plus complexe obéissant à un protocole spécifique.

Les problématiques auxquelles nous cherchons à trouver des solutions sont :

Comment modéliser le scénario d'intégration des processus métiers ? Quels sont les modèles et les formalismes qui seront utilisés ?

Les approches d'intégration d'applications tirées de la littérature, présentent le problème de la cohérence des processus. En effet, il n'est pas sûr qu'une application, basée sur des services distants et interconnectés par des réseaux, aboutira par la bonne intégration des processus à une application correcte. Pour cela, il est nécessaire d'avoir une sémantique opérationnelle précise des langages de description comportementale de ces processus.

Comment intégrer les processus métiers au moment d'exécution ? Comment améliorer la gestion et le suivi de cette intégration ?

Le comportement coopératif des processus métiers résulte d'un modèle d'interaction fixé au préalable par le concepteur du système. En effet, l'exploitation de ce modèle au moment d'exécution est une tâche indispensable. Ce modèle permet aux agents de coordonner les processus métiers (qui sont décrits dans ce modèle) pour atteindre un objectif commun.

Une solution possible pour remédier à ces problèmes consiste à utiliser conjointement les services Web et les SMA. Dans la section suivante, nous illustrons ce couplage (service Web et agent) par une présentation générale de notre approche.

3 Objectifs et contributions de la thèse

Notre travail se situe au carrefour des recherches sur le Génie Logiciel, l'intégration d'applications par les processus et les SMA. Plus précisément, ce travail de recherche concerne l'intégration d'applications par les processus à l'aide de systèmes multi-agents dits communiquants, c'est à dire les agents interagissent par l'envoi de messages, en utilisant un langage de communication de haut niveau.

Dans ce contexte, l'intégration d'applications se focalise sur **les Protocoles d'Interaction (PI)** [5] qui constituent l'unité fondamentale de la modélisation.

Notre travail se situe à deux niveaux (voir la figure 2) :

- Niveau conceptuel : la modélisation et la vérification du scénario d'intégration (niveau conceptuel de la figure 2).
- Niveau opérationnel : l'exploitation et la gestion de l'interaction au moment de l'exécution (niveau opérationnel de la figure 2).

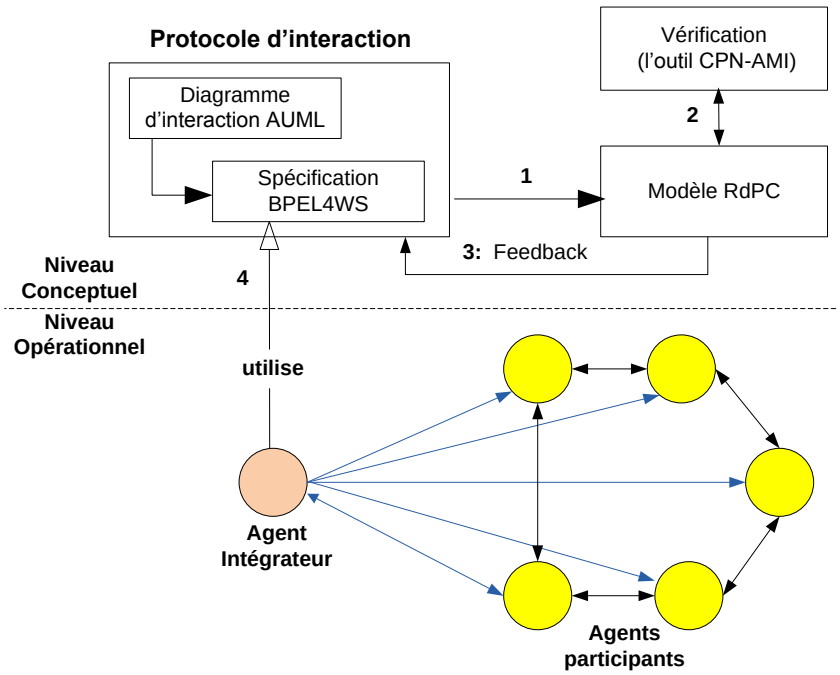

Figure 2 – *Vue globale de l'approche proposée*

5. Dans le reste de cette thèse, nous utilisons l'acronyme PI pour Protocole d'Interaction

Les protocoles d'interaction : une abstraction du scénario d'intégration

Dans cette thèse, nous introduisons l'idée de conception orientée *interaction* pour l'intégration des processus métiers [25].

Le concept des protocoles d'interaction nous permet de définir les spécifications du protocole (i.e. l'ensemble des fonctionnalités et des propriétés qui doivent être présentes dans le scénario d'intégration des processus). L'ensemble des informations collectées lors de cette phase sont les processus métiers et les services Web intervenant dans le scénario d'intégration, les prérequis nécessaires et les contraintes imposées à l'exécution du protocole.

Nous utilisons le formalisme graphique AUML [6] [10] accompagné d'une spécification textuelle BPEL4WS [7][64] pour la modélisation des actions de communication entre les différents partenaires de l'interaction. AUML est une méthode de représentation graphique des protocoles d'interactions au niveau des "performatifs" [8]. Alors que BPEL4WS est un langage de composition de services Web permettant de définir des processus métiers et de décrire les interaction entres les services. BPEL4WS décrit ces interactions à travers un plan en spécifiant les flots de contrôle entre les services partenaires et les dépendances des données entre plusieurs processus métiers.

Nous proposons par la suite, une sémantique opérationnelle de notre modèle (la flèche 1 de la figure 2). Cette sémantique est basée sur des règles de transformation garantissant le respect de la cohérence du système. Le modèle formel retenu pour la représentation du comportement observable des processus métiers est les réseaux de Petri colorés (RdPC) [14]. Nous avons aussi développé l'outil *IP2CPN* pour la génération automatique des RdPC.

Pour la vérification de protocole d'interaction mis en oeuvre (la flèche 2 de la figure 2), nous utilisons l'environnement logiciel CPN-AMI[23] développé au LIP6 [9]. Cet environnement supporte la spécification et la vérification de RdP. Ses outils permettent de vérifier des formules de logique temporelle (*model checking*) et de calculer des propriétés structurelles (invariants, verrous, trappes)[4].

Dans notre travail, nous nous intéressons à deux types de propriétés. Le premier concerne des propriétés générales que l'on retrouve dans tous les protocoles comme les problèmes d'interblocage et de réception non spécifiée. Le second porte sur des propriétés fonctionnelles qui dépendent

6. Agent Unified Modeling Language
7. Business Process Execution Language for Web Services
8. Les actions intentionnelles effectuées au cours d'une communication
9. Laboratoire d'Informatique de Paris 6

de la fonction du protocole et sont décrites dans le cahier des charges lors de la phase d'analyse.

Une architecture basée agent pour l'intégration d'applications

Les phases précédentes ont permis de concevoir un protocole formalisé et validé. En se fondant sur ce protocole d'interaction, nous définissons une architecture système permettant l'intégration des processus métiers (voir le niveau opérationnel de notre approche dans la figure 2). Cette architecture repose sur deux éléments clés : un framework modulaire d'agents interactifs et une bibliothèque de composants réutilisables.

Le framework d'agents fournit les composants de base et une structure minimale d'agents interactifs que le développeur réutilisera pour développer ses agents.

La bibliothèque contient un ensemble de protocoles d'interaction et elle est exploitée par un agent intermédiaire baptisé "Agent Intégrateur".

L'idée de créer un agent intermédiaire pour exécuter les protocoles d'interaction offre beaucoup d'avantages, notamment, la réutilisation et la maintenance des protocoles d'interaction. En plus, elle évite d'encombrer les agents participants avec les règles d'interaction.

4 Organisation de la thèse

Le manuscrit est structuré en quatre chapitres et une conclusion générale.

Le premier chapitre est consacré à une étude bibliographique de l'intégration d'applications et des systèmes d'entreprise dans le cadre général. Nous montrons la diversité des définitions établies pour l'interopérabilité et l'intégration selon les différents domaines et les différents systèmes impliqués. Nous présentons par la suite quelques concepts de l'approche agent et ses apports dans la mise en oeuvre de l'intégration d'applications.

Nos contributions démarrent au chapitre 2. Dans ce chapitre, nous proposons une « approche orientée interaction » pour la modélisation de l'IAE. Cette approche permet de pallier les limites des techniques de l'intégration d'applications par coordination des processus car ils apportent des solutions et des modèles pour tenir compte de plusieurs contraintes telles que la répartition des partenaires, l'ouverture et la flexibilité de leur environnement.

Le troisième chapitre est consacré à notre deuxième contribution : une architecture basée agent pour l'IAE. Il présente les motivations pour le choix de l'approche agent, la spécification de l'architecture en termes de structures d'agents, de comportements des agents, ainsi que la communication inter-agents.

Le chapitre 4 illustre l'application de notre approche à une étude de cas. Nous décrivons tout d'abord les différentes étapes et les modèles utilisés dans notre travail ainsi que l'architecture proposée. Nous précisons ensuite la plate-forme d'implémentation et comment les concepts de base proposés par notre approche (spécification d'agents, interactions d'agents, etc.) peuvent être réaliser via cette plate-forme.

Enfin, le manuscrit se termine par une conclusion générale qui récapitule les travaux réalisés et propose quelques visions pour les travaux futurs.

État de l'art

1

Sommaire

Ce chapitre est consacré à un état de l'art sur les travaux liés à l'intégration d'applications d'entreprise et ceux relatifs à l'interaction dans les systèmes multi-agents. Nous les présentons en deux parties.

Dans une première partie, nous proposons une catégorisation selon laquelle nous décrivons et analysons les approches existantes dédiées à l'intégration d'applications. Dans la seconde partie, nous présentons un panorama des modèles et des langages pour la formalisation des protocoles

d'interaction.
Nous concluons ce chapitre par les limites que les travaux existants présentent pour la modélisation du problème de l'IAE.

1.1 Intégration d'applications et processus métiers

L'intégration des systèmes d'information constitue de nos jours l'un des problèmes complexes auxquels l'entreprise est confrontée. Au coeur de cette problématique se pose entre autres le problème de l'interopérabilité. Les problèmes d'intégration et de l'interopérabilité sont devenus aujourd'hui cruciaux car les applications composant les systèmes d'information d'entreprise ont besoin de plus en plus de travailler ensemble [66]. Cela peut être dans le cadre intra-entreprise où il s'agit principalement d'interconnecter diverses applications hétérogènes d'une même entreprise, ou dans le cadre inter-entreprise pour faire communiquer des applications de plusieurs partenaires, ou encore dans le cadre des logiques de fusion-acquisition d'entreprises pour unifier les systèmes d'information impliqués.

La section suivante a pour objet de présenter la problématique générale de l'intégration et de l'interopérabilité d'applications d'entreprises.

1.1.1 Concepts d'intégration et d'interopérabilité

Les problèmes d'intégration et d'interopérabilité ne sont pas nouveaux et ils constituent des domaines complexes auxquels s'intéressent plusieurs communautés dont celle des systèmes d'information, celle des bases de données, celle du génie logiciel, celle de la fouille de données, etc. Plusieurs approches, standards et/ou technologies ont été proposés pour résoudre ces problèmes. Dans cette section, nous allons présenter quelques notions générales de ces concepts.

Le concept d'intégration

Selon le dictionnaire, l'intégration désigne l'action d'intégrer. De son étymologie latine "integrare", intégrer signifie "rendre entier" ; faire entrer dans un ensemble, dans un groupe plus vaste. Il s'agit donc de faire tomber les barrières fonctionnelles et organisationnelles au sein des entreprises afin que l'ensemble soit vu comme un tout cohérent.

De façon générale, l'intégration consiste à combiner des composants de façon à former un nouvel ensemble constituant un tout pour créer de la synergie [135]. Dans le contexte de l'entreprise, le problème d'intégration est communément appelé intégration d'entreprise. Vernadat définit l'intégration d'entreprise comme étant le processus qui "consiste à faire interopérer fortement les personnes, les machines et les applications afin d'accroître la synergie au sein de l'entreprise" [131].

Dans le même ordre d'idées, Williams définit l'intégration d'entreprises comme "la coordination de tous les éléments incluant les processus

métier, les ressources humaines, et la technologie d'une entreprise fonctionnant ensemble dans le but d'atteindre la réalisation optimale de la mission de cette entreprise telle que définie dans la stratégie de l'entreprise" [136].

Le concept d'interopérabilité

L'interopérabilité se focalise principalement sur les aspects techniques. Elle concerne à la fois les systèmes matériels et les systèmes logiciels [131] [96]. De nombreuses définitions existent dans la littérature (nous en citons les plus communes) :

- L'interopérabilité est la capacité des systèmes informatiques et des processus qu'ils supportent d'échanger des données et de permettre la partage d'information et de connaissance [37] ;
- L'interopérabilité est le moyen par lequel les systèmes, les informations et les méthodes de travail sont interconnectés : à l'intérieur des administrations ou entre ces dernières, au niveau national ou à travers toute l'Europe, ou avec les entreprises [38] ;

A ce stade, Meinadier [88] précise que l'interopérabilité de systèmes se place ainsi au niveau du partage d'informations et de services et implique des adaptations, notamment sémantiques, entre les modèles de données, tandis que l'intégration entre systèmes se place au niveau du comportement global : elle implique, de plus, l'intégration fonctionnelle et l'enchaînement des traitements entre les systèmes d'information. Cette précision rejoint celle de Vernadat [131] qui considère aussi le concept d'intégration comme un concept plus large que celui de l'interopérabilité dans la mesure où il englobe, selon lui, des capacités liées à la communication, la coopération et à la coordination des composants du système.

Dans le cadre de nos travaux, nous considérons que l'intégration est un concept générique incluant le concept d'interopérabilité tout en nous focalisons sur le concept d'intégration.

1.1.2 Concept de processus métier

Le terme de « processus métier » est souvent utilisé pour désigner des notions différentes : processus abstrait, processus exécutable, ou processus collaboratif [133].

Un processus métier est une chorégraphie d'activités incluant une interaction entre participants sous la forme d'échange d'informations [128]. Les participants peuvent être :

- Des applications / services du système d'information,
- Des acteurs humains,
- D'autres processus métiers.

Par conséquent, un processus métier est appréhendé comme un ensemble d'activités, pouvant être considérées comme une succession de transformations. Son déroulement peut être indiqué par des transitions, éventuellement caractérisées par des événements. L'agencement des activités correspond à la structure du processus [30].

Un processus métier peut être interne à une entreprise, ou peut mettre en jeu des entreprises partenaires. Dans ce dernier cas, on parle alors de processus public ou collaboratif.

Processus public ou collaboratif

Ce type de processus incluant n partenaires est composé de deux parties : une interface et n implémentations. L'interface définit la partie visible du processus, c'est-à-dire le contrat entre les partenaires : définition des documents métiers échangés, du séquencement des activités, des rôles et responsabilités de chaque partenaire. L'exécution spécifique de chaque partenaire est abstraite grâce à cette interface.

Processus exécutable

Un processus peut être rendu exécutable de trois façons [128].

- Il peut orchestrer les applications / services du système d'information ainsi que des actions utilisateurs pour rendre une tâche automatisée. Par exemple, un processus de gestion de bons de commande reçoit les bons de commande via des messages XML, les transmet aux personnes adéquates, se renseigne sur la disponibilité des éléments commandés dans les bases de données de l'entreprise, etc.
  Par conséquent, rendre un processus exécutable ne signifie pas nécessairement l'automatiser. Par exemple un processus peut uniquement être l'automatisation de la transmission d'informations entre acteurs (approche Workflow), les actions étant effectuées manuellement par les utilisateurs.
- En outre, rendre un processus exécutable peut aussi signifier introduire des points de contrôle pour permettre le contrôle de son déroulement. On parle alors de processus de BAM [1].
- Le troisième cas consiste à utiliser les nouveaux standards de Web. L'exemple le plus récent est celui des services Web : les éditeurs proposant des solutions concurrentes coopèrent désormais pour la définition des spécifications permettant à leurs outils de communiquer[75]. Parmi les langages de descriptions et d'execution des processus métiers , nous citons XLANG [117], WSFL [47] et BPEL4WS [64].

1. Business Activity Monitoring

1.1.3 Différents types d'intégration d'applications

La plupart des travaux tirés de la littérature sur l'intégration d'applications permettent de mettre en évidence principalement quatre approches qui sont : l'intégration des applications par les données, les traitements, les présentations et par les processus. Les trois premières approches peuvent être considérées comme le résultat logique de la structuration des applications en couches, tandis que la quatrième approche, qui se décline par la suite en une combinaison des trois approches de bases, résulte de la mise en oeuvre d'un médiateur (appelé moteur de workflow), permettant ainsi d'interconnecter des applications via l'orchestration des processus.

La notion de processus joue un rôle majeur dans la maîtrise de l'évolution des systèmes informatiques associés. En effet, à cause de la diversité et de la plasticité des technologies de l'information, l'efficacité des systèmes installés dépend de la cohérence entre la stratégie, l'organisation et les systèmes d'information [113] [54]. Dans cette perspective d'alignement stratégique, la mission première du système d'information est d'aider l'entreprise à atteindre ses objectifs, sans être dominée par des contraintes techniques.

Pour que le système informatique puisse évoluer de façon réactive et efficace, les informaticiens doivent le construire de façon modulaire, comme un assemblage de composants faiblement couplés, permettant de modifier chaque composant de façon quasi indépendante. Par ailleurs, les gestionnaires sont appelés à devenir acteurs de leur système d'information et des services qu'il apporte, et à le structurer de façon à ce qu'il soit flexible. Cela a conduit à définir une notion d'architecture de système d'information avec trois vues [69][77] :

1. La vue métier, qui pilote les deux autres, est celle des activités de l'entreprise, c'est-à-dire ses processus. Une telle représentation est appelée *architecture métier* ou une cartographie des processus.
2. La seconde vue, souvent appelée *architecture fonctionnelle*, fait apparaître les fonctions et les informations du système d'information.
3. La troisième vue appelée *architecture applicative*, est celle du système informatique, c'est-à-dire des composants logiciels, des applications et des infrastructures techniques.

L'un des enjeux actuels de la maîtrise des systèmes d'information d'entreprises est de construire une architecture métier et une architecture fonctionnelle supportant l'interopérabilité de plusieurs processus métiers afin d'accroître la synergie entres les applications d'entreprises [12]. Une des pistes peut être l'approche d'intégration des processus métiers que nous proposons dans cette thèse, après avoir effectue un état des pratiques de modélisation existantes.

L'intégration d'applications par les processus (ou BPI [2]) recouvre la définition et la gestion des échanges d'information d'une entreprise à travers une sémantique des processus offrant une meilleure vue d'ensemble plus qu'une simple analyse du parcours des données échangées. La BPI définit comment est gérée une séquence d'événements ; ces événements représentent des activités plus ou moins longues devant être complétées pour traiter une tâche faisant partie d'un processus métier. Ces activités consistent surtout à récupérer, à modifier et à actualiser les données des systèmes informatiques. Afin d'implémenter ces activités, la BPI consisterait idéalement à pouvoir définir un modèle graphique du processus métier et à générer automatiquement l'intégration entre modèle, systèmes informatiques et acteurs [48] [17].

Malheureusement, la réalité de la BPI est tout autre. La plupart des produits BPI sont incapables de générer les parties plus complexes de l'implémentation de l'intégration, telles que la création de connexions aux systèmes informatiques. Néanmoins il est possible de générer à partir d'un modèle de processus métiers des alignements (mappings) avec les données que le processus manipulera. Cependant, cela ne représente en général pas plus de 50% du travail nécessaire pour compléter l'intégration [48]. Pour compléter cette intégration, il faut utiliser des technologies telles que le workflow, les serveurs d'applications, l'Intégration d'applications d'Entreprises (IAE) et les services Web (Fig. 1.1).

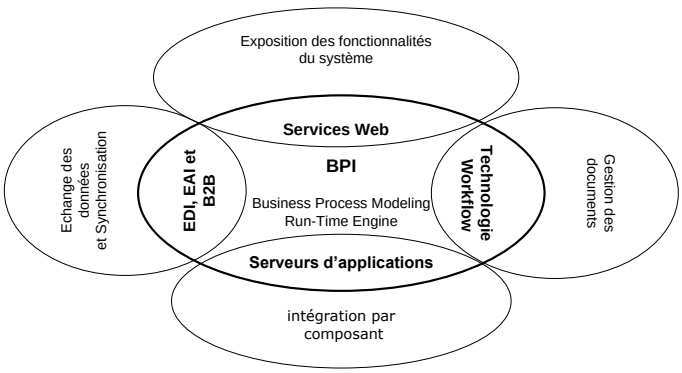

FIGURE 1.1 – *Relations entre le workflow, les serveurs d'applications, l'IAE et les Services Web*

1.1.4 Quelques approches pour l'intégration d'applications

Dans cette section, nous présenterons quelques technologies d'intégration d'applications en soulignant la manière dont elles permettent des connexions faciles aux applications d'entreprises. Chaque partie compor-

2. En anglais BPI pour *Business Process Integration*

tera une présentation de la technologie et ses principaux composants. Ensuite leurs avantages et leurs limites seront présentés.

Workflow et serveurs d'application

Le workflow et les serveurs d'applications sont deux composantes importantes pour l'intégration des processus métiers : la première gère la circulation des documents échangés par les différents protagonistes impliqués dans le processus, tandis que les serveurs d'applications fournissent les ressources dont on a besoin pour traiter les processus répartis [7]. Dans cette partie, nous verrons que même s'ils n'ont pas les mêmes buts, le workflow et les serveurs d'applications partagent la même évolution.

Le workflow peut être défini comme étant " l'automatisation, totale ou partielle, d'un processus métier, dans lequel les tâches passent d'un participant à l'autre pour être effectuées, selon un ensemble de règles procédurales ". Il organise les processus métiers de deux façons : il définit l'ordre dans lequel les tâches doivent être accomplies, et stipule les étapes et les points de contrôle où interviennent des êtres humains.

Les solutions workflow offrent en général les avantages suivants :

- Reporting : Ils gèrent les processus en mesurant leur durée, leur taux de réussite, etc.
- Identification des problèmes : Ils fournissent en cas de problème un mécanisme d'alerte au gestionnaire du processus.
- Simulation : Ils offrent un outil de simulation pour aider l'entreprise à prendre des mesures afin de continuellement améliorer ces processus

La force des solutions workflow réside dans leur grande faiblesse, elles sont conçus pour travailler avec des humains (c'est-à-dire avec les documents qu'ils ont créés) et non pas avec des systèmes d'information de l'entreprise. En d'autres termes, ils n'ont pas étés conçus pour intégrer des applications à un processus [48][17].

D'un autre coté, les serveurs d'applications sont utilisés pour centraliser les processus des applications sur des machines plus puissantes. Ils fournissent des outils et un environnement pour développer et faire usage d'applications réparties.

En outre, les serveurs d'applications deviennent de plus en plus importants pour l'intégration des processus métiers des entreprises, car ils présentent les avantages suivants :

- Ils offrent un mécanisme d'intégration par composants, c'est à dire que les composants écrits avec des langages différents sont compatibles via des interfaces standards telles que EJB [3] ou CORBA [4] [116].

3. Enterprise JavaBeans
4. Common Object Request Broker Architecture

– Ils offrent des services relatifs au temps d'exécution. Ces services recouvrent la gestion des transactions, celle des objets, celle des sessions et celle des événements et des pannes.
– Ils offrent une plateforme fiable, cette dernière offre des services comme la gestion des transactions particulièrement adaptés à l'IAE et à ses propres services comme le routage intelligent des messages.

Les serveurs d'applications ont tendance à être excessivement complexes à utiliser. La plupart des constructeurs, comme IBM avec *WebSphere* [35], offrent d'ailleurs une palette de services afin d'aider les entreprises à optimiser leur utilisation. Et, comme c'est le cas pour des solutions workflow, il manque aux serveurs d'applications, des fonctions nécessaires à l'intégration d'applications comme la transformation des formats et le routage intelligent des données.

Les applications basées sur un échange des messages

Ce type d'applications est apparu assez récemment et utilise les technologies de messages inter-applications (ou MOM pour Message Oriented Middleware). En général, les solutions MOM fonctionnent en faisant circuler les messages d'une manière asynchrone d'un programme à un autre ou à plusieurs programmes. Contrairement à ORB [5] et RPC [6], MOM considère que la couche de transport n'est pas fiable, comme cela se produit lorsque les programmes doivent communiquer via un réseau local ou Internet.

Dans le cadre de l'intégration d'applications, un type particulier de MOM, connu sous le nom de pub/sub (publier/souscrire), convient le mieux à la BPI. Avec pub/sub, les programmes souscrivent à un sujet défini par le développeur [17].

Dans ce cas, les programmes peuvent aussi publier aussi des messages relatifs à un sujet donné. Une fois qu'un programme a souscrit à un sujet, le programme recevra, en temps réel, tous les messages publiés sur le sujet en question. Ce système est particulièrement adapté à la BPI étant donné qu'il permet la livraison de messages en temps réel à un grand nombre de clients. Cela est particulièrement utile dans un cadre de collaboration où l'automatisation du mouvement des données entre les applications et des processus métiers demande une livraison immédiate des messages.

Mais pub/sub n'est pas toujours la solution idéale. Dans le contexte de l'IAE, il existe certains appréhensions sur son efficacité en tant que solution au système d'envoi des messages. Un autre problème avec MOM est celui de l'interopérabilité [114]. En effet un MOM est généralement implé-

5. Object Request Broker
6. Remote Procedure Call

menté comme un protocole propriétaire de bas niveau, ce qui signifie que l'implémentation est souvent incompatible avec d'autres implémentations.

Adaptateurs et Connecteurs

Dans le cadre de l'IAE, il y a deux façons d'intégrer les applications : une intégration invasive et une intégration non invasive. Une intégration invasive (c'est-à-dire une modification des interfaces) n'est ni souhaitable ni possible avec un grand nombre de systèmes d'origine et de progiciels. Pour cette raison, une intégration non invasive constitue la façon la plus courante d'aborder le problème. Néanmoins, lorsqu'on veut connecter deux systèmes, l'API[7] du premier système doit au moins être adapté pour convenir au modèle de programmation de l'autre système, et les données doivent être transformées pour être échangées entre les deux modèles de données. Par conséquent, nous avons besoin d'adaptateurs et de connecteurs [17] [72].

Dans la plupart des solutions IAE, il existe des adaptateurs et des connecteurs pour les applications les plus couramment utilisés en entreprise telles que celles de SAP, Siebel et Oracle. Parce que ces connecteurs sont pré-installés, ils permettent aux applications d'être connectées rapidement, nécessitant ainsi un effort de développement minime. Mais tous les adaptateurs/connecteurs ne peuvent pas être pré-installés pour tous les systèmes disponibles : les entreprises doivent souvent développer leurs propres adaptateurs et connecteurs. Pour ce faire, elles auront peut-être besoin de concevoir l'application de façon à ce qu'elle puisse gérer elle-même les services comme la sécurité, les transactions, la gestion des connexions, etc. Cela prend beaucoup de temps, coûte cher et en conséquence le travail est souvent bâclé [48].

1.1.5 Synthèse

Dans cette section nous avons vu qu'une sélection faite avec soin des composants principaux d'une solution IAE pouvait conduire à un couplage inférieur lors de l'intégration entre systèmes. Néanmoins, nous remarquons la difficulté et le temps qu'il faut pour :

- mettre en place les systèmes de messagerie.
- développer les adaptateurs d'applications requis.

De nos jours, les entreprises qui utilisent les solutions IAE sont conçues pour relier durablement, via des connexions discrètes, des applications souvent monolithiques. Mais pour la gestion des processus métiers, les entreprises ont besoin de technologies qui soient très flexibles et capables

7. API : Application Programme Interfece

d'intégrer rapidement, des systèmes à travers toutes sortes de barrières technologiques, quelle que soit la durée d'existence des liens. Pour cette raison, dans la section suivante, nous étudierons les services Web, une collection de standards basés sur XML, fournissant un moyen modulaire et dynamique de transmettre l'information entre les applications.

1.2 Les services Web

Les services Web sont des composants logiciels qui peuvent être décrits, publiés, localisés et invoqués sur un réseau. En d'autres termes, les services Web procurent une méthode standardisée (utilisant des formats XML) pour publier et souscrire à des services logiciels via des protocoles Web tels que HTTP [49]. On peut considérer la technologie services Web comme étant l'évolution d'une technologie d'intégration telle que CORBA (Fig. 1.2).

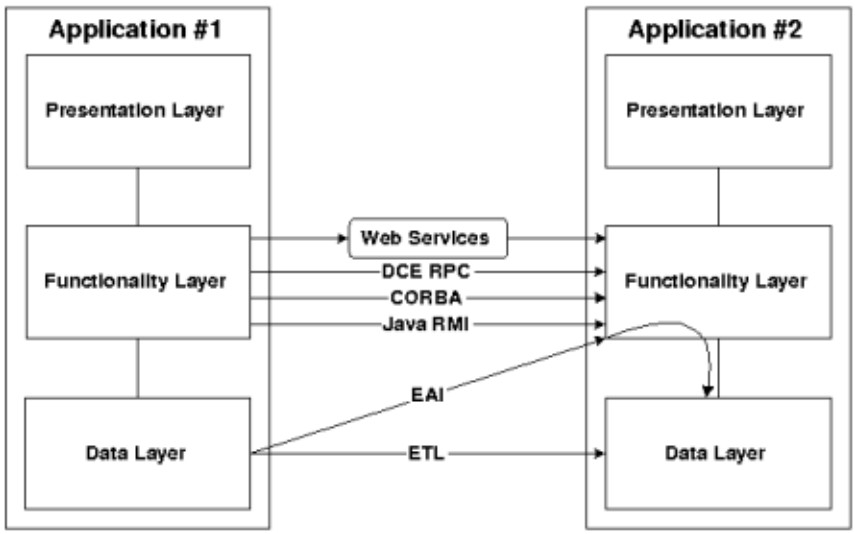

Figure 1.2 – *Principales méthodes d'intégration d'applications [107]*

1.2.1 Principes de l'architecture orientée services

Une architecture orientée services consiste essentiellement en une collection de services qui interagissent et communiquent entre eux. Cette communication peut consister en un simple retour de données ou en une activité (coordination de plusieurs services).

Un service est une entité de traitement qui respecte les caractéristiques suivantes [97] :

Large Granularité (*coarse-grained*) : Les opérations proposées par un

service encapsulent plusieurs fonctions et opèrent sur un périmètre de données large, contrairement à la notion de composant technique.

Exposition d'un contrat d'utilisation : Un service expose un contrat d'utilisation décrit en deux parties. Une partie abstraite qui déclare les messages d'entrée et de sortie du traitement offert. Une autre partie, qui est concrète décrit les standards et protocoles techniques utilisés pour l'activation du service. Selon les choix d'implémentation et de déploiement, il est possible d'avoir plusieurs parties concrètes pour une même partie abstraite. On nomme aussi le contrat d'utilisation par le terme d'interface de service.

Localité : Avant d'appeler (bind, invoke) un service, il faudra le trouver (find).

Couplage faible (*loosely-coupled*) : Les services sont connectés aux clients et autres services via des standards. Ces standards assurent le découplage, c'est-à-dire la réduction des dépendances. Ces standards sont des documents XML comme dans les services Web.

1.2.2 Spécifications de services Web

Bien plus qu'une nouvelle technologie d'intégration, les services Web représentent une pile de spécifications émergentes (Fig. 1.3) dont le but est de définir une architecture modulaire orientée services.

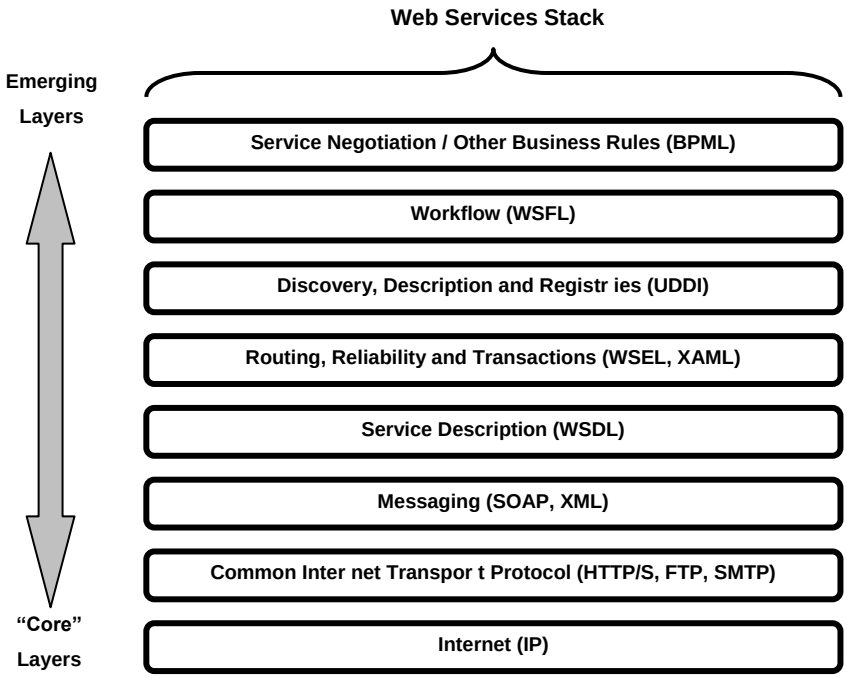

FIGURE 1.3 – *Les spécifications des services Web [107]*

Au bas de la pile se trouvent les spécifications définissant l'infrastructure permettant un accès dynamique aux services Web. Ces spécifications concernent la définition des services Web (WSDL [8]), enregistrement et localisation (UDDI [9]) et invocation des service Web (SOAP [10]).

- SOAP [91] est un protocole de type RPC [11] utilisant XML pour définir ces messages et les envoie utilisant les protocoles standard Internet tels que HTTP. C'est une manière simple d'échanger des données dans un environnement réparti tel que le Web. Elle permet à deux points finaux de se connecter préalablement à la demande d'utilisation d'un service. Le message SOAP est codé en XML et transmis en utilisant HTTP. Il transporte les données échangées entre les deux objets éloignés et les données nécessaires à l'exécution du service sollicité : nom de la méthode, type de données, etc.
- Avec WSDL [34], les services (par exemple, les objets dans Java, C++ ou COM), peuvent être décrits de façon à ce que leurs futurs utilisateurs sachent comment les activer et où les trouver. Par analogie avec CORBA, WSDL correspond la définition de l'interface (IDL).
- UDDI [124] vient compléter l'ensemble de ces technologies en fournissant un répertoire (public ou privé) où les prestataires peuvent enregistrer leurs services et les utilisateurs peuvent venir les récupérer.

Plus haut dans la pile se trouvent les spécifications dont le but est de fournir une fonctionnalité ajoutant un contexte et des objectifs aux spécifications présentées plus haut. Par exemple, les spécifications pour un modèle de processus métiers seront utilisées dans un système de gestion des processus métiers pour faire en sorte que les processus (publiés comme Web Services) puissent être orchestrés de façon cohérente. Mais, contrairement aux spécifications essentielles qui sont presque considérées comme étant standards, il existe des approches divergentes (comme Business Process Modelling Language (BPML [12]) de *BPMi.org* en concurrence avec Business Process Execution Language for Web Services (BPEL4WS [13]) de IBM et Microsoft.

1.2.3 Apports et limites des services Web dans l'intégration d'applications

Les services Web permettent de promouvoir l'utilisation d'une intégration centrée sur les services, où les fonctions automatiques sont mises à la

8. Web Services Description Language
9. Universal Description Discovery and Integration
10. Simple Object Access Protocol
11. Remote Procedure Call
12. Business Process Modeling Language
13. Business Process Execution Language for Web Services

disposition de tous les utilisateurs éventuels, quelles que soient la technologie et les spécifications des interfaces qu'ils utilisent [105]. Les services Web ont en outre les avantages suivants :

- Ils supportent non seulement le système synchrone d'envoi de messages de RPC, mais également le système asynchrone, rendant ainsi possible une architecture asynchrone couplée sans rigidité, adaptée aux transactions commerciales durables, qui peut encore fonctionner sans réponse immédiate du réseau.
- Ils évitent le principal problème des messageries IAE, à savoir leur dépendance vis-à-vis de solutions propriétaires. Ils le font en se basant sur des standards comme HTTP, XML et WDSL. Les solutions MOM, au contraire, sont par exemple souvent liées à la plateforme de l'éditeur de logiciels.

Ainsi, les approches proposées pour l'intégration d'applications visent à satisfaire les cas d'usage précités (i.e. les procédures génériques) par l'orchestration ou l'enchaînement de processus métiers [92]. Dans ces approches, un schéma d'orchestration, spécifié par un expert, est associé à chaque procédure générique à satisfaire. Chaque schéma regroupe les profils ou les identifiants des services à composer, ainsi que leur enchaînement (ex. séquence d'appel des actions des services). Lorsque l'utilisateur requiert l'exécution de l'une de ces procédures génériques (ex. organisation d'un voyage), le schéma d'orchestration donné a priori est instancié en fonction des services concrets disponibles et ensuite exécuté [21].

Il est tout de même à noter que ces approches présentent plusieurs limites dont voici les trois plus récurrentes :

- Un service web est passif jusqu'à ce qu'il soit invoqué ;
- Un service web a seulement connaissance de lui-même, mais pas de ses applications ou utilisateurs clients ;
- Un service web n'est pas adaptable, et il n'est pas capable de bénéficier des nouvelles capacités de l'environnement afin d'apporter des services améliorés

L'autonomie, l'adaptation et la coopération, points faibles des services Web, sont par ailleurs des mécanismes qui ont largement été explorés dans plusieurs travaux de recherche effectués dans le domaine des systèmes multi-agents. Plusieurs études décrivent à ce propos la pertinence de la combinaison des technologies agent et services Web [26][21][89].

Dans ce qui suit, nous faisons une brève présentation des systèmes multi-agents et du mécanisme d'interaction qu'ils intègrent. Nous rapprochons les domaines de l'intégration d'applications et de la coordination multi-agents avant d'esquisser notre proposition pour la définition d'une approche conceptuelle et technique pour l'intégration d'applications par coordination des processus métiers.

1.3 Systèmes multi-agent et Interaction

Les SMA étudient la manière de répartir un problème sur un certain nombre d'entités coopératives. Elles s'intéressent à la manière de coordonner le comportement intelligent d'un ensemble d'entités selon des lois sociales. Ces entités ou agents sont autonomes et interagissent dans un environnement pour la résolution de problèmes (leur comportement dans une société d'agents)[70].

1.3.1 Concept d'agent

Le concept d'agent a été l'objet d'étude pour plusieurs décennies dans différentes disciplines. Il a été non seulement utilisé dans les systèmes à base de connaissances, la robotique, le langage naturel et d'autres domaines de l'intelligence artificielle, mais aussi dans des disciplines comme la philosophie et la psychologie. Aujourd'hui, avec l'avènement de nouvelles technologies et l'expansion de l'Internet, ce concept est encore associé à plusieurs nouvelles applications comme agent ressource, agent courtier, assistant personnel, agent interface, agent ontologique, etc [40]. Dans la littérature, on trouve une multitude de définitions d'agents. Elles se ressemblent toutes, mais diffèrent selon le type d'application pour laquelle est conçu l'agent. A titre d'exemple, voici l'une des premières définitions de l'agent dûe à Ferber [43] :

> "Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents."

Il ressort de cette définition des propriétés clés comme l'autonomie, l'action, la perception et la communication. D'autres propriétés peuvent être attribuées aux agents selon une autre définition du concept d'agent proposée par Jennings, Sycara et Wooldridge [110] :

> "Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu."

Les notions "situé", "autonomie" et "flexible" sont définies comme suit :

situé : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement.

autonome : l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.

flexible : l'agent dans ce cas est :

- capable de répondre à temps : il doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis,
- proactif : il doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au "bon" moment,
- social : il doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

Notons que, dépendants des applications, certaines propriétés sont plus importantes que d'autres. Il peut même s'avérer que pour certains types d'applications, des propriétés additionnelles sont requises. Il convient cependant de souligner que la présence des propriétés comme l'autonomie, la flexibilité, la sociabilité, etc., donne naissance au paradigme agent tout en le distinguant des systèmes conventionnels comme les systèmes répartis, les systèmes orientés objets et les systèmes experts.

1.3.2 Interactions et modes d'interaction

Dans le cadre de SMA, le concept d'interaction est lié à celui de communication. Celle-ci y joue un rôle important : elle assure le transfert d'informations entre les agents du système et par la même constitue une action élémentaire [1]. En effet, les communications dans les systèmes multi-agents sont à la base des interactions et de l'organisation des agents. Nous distinguons essentiellement deux modes de communication : la communication indirecte qui est effectuée par signaux via l'environnement, et la communication directe qui procède à un échange de messages entre les agents.

Le mode de communication par échange de messages entre les agents provient initialement du domaine des acteurs. Il permet à un agent de planifier un acte de communication envers un autre agent, on parle alors de communication intentionnelle. Cette approche est inspirée de l'interaction sociale telle que nous la trouvons dans d'autres contextes comme la communication entre les humains. A la différence des autres approches, celle-ci a donné lieu à des théories calculatoires formelles permettant de savoir comment communiquer incrémentalement ou comment ajuster un plan de communication. Ceci est fait dans le but de s'adapter à un monde incertain, où les circonstances changent, rendant ainsi possible la communication complexe qui consiste en l'échange d'informations stratégiques.

L'interaction inter-agents regroupe et combine plusieurs types de messages. Cette combinaison se traduit par des enchaînements conversationnels qui peuvent être modélisés par deux approches [43] :

– Dans la première approche, l'agent construit son propre plan de

communication au moment où il en a besoin pour réaliser sa tâche. Ainsi, l'interaction n'est pas définie a priori, elle est émergente. Les connaissances et les buts des agents régissent l'interaction en spécifiant le message à envoyer (acte de communication, agents destinataires, le contenu du message,...). Cette approche donne à l'agent plus de flexibilité dans son interaction. Cependant, dans cette approche l'agent doit avoir suffisamment de connaissances sur la sémantique des messages et qu'il doit être doté d'un mode de raisonnement lui permettant de mener ses interactions.

– Dans la deuxième approche, l'interaction se conforme à un enchaînement de messages spécifié à l'avance. Les règles qui régissent l'échange de messages forment le protocole d'interaction. De plus, le protocole d'interaction définit le type des messages qui doivent être échangés. L'agent interactif doit se conformer aux règles de conversation dictées dans le protocole. Comparée à la première approche, l'interaction basée sur les protocoles d'interaction ne nécessite pas d'architecture complexe au sein de l'agent. Ainsi, un protocole d'interaction spécifie un ensemble limité de réponses possibles pour un type spécifique de messages. Cette approche d'interaction, basée sur des protocoles bien définis, est celle qui sera considérée dans la section suivante.

1.3.3 Théorie des actes de langage

Le typage sémantique des messages prend son origine de la théorie des actes de langage. Cette théorie de la philosophie du langage est apparue dans les années 60 par les travaux de Austin [9] et ensuite de Searle [118]. Elle part du principe que la communication humaine est constituée d'actes de langage, aussi appelé performatifs, que l'on peut classer en cinq grandes catégories :

– Les assertions, qui informent l'interlocuteur d'une croyance de la part du locuteur sur l'état du monde ;
– Les directives, qui sont des requêtes, indiquant l'intention du locuteur que l'interlocuteur agit de façon à satisfaire la requête ;
– Les actes expressifs, exprimant des sentiments du locuteur ;
– Les déclarations, qui informent l'interlocuteur d'une intention du locuteur ou un état de quelque chose qui dépend de ce dernier ;
– Les engagements, qui sont des promesses, ou à l'inverse des menaces.

Cette théorie a ensuite été appliquée à la communication entre agents autonomes, par Cohen et Levesque [106], en partie par anthropomorphisme, puisque les actes de langages sont à l'origine un modèle philo-

sophique de la communication humaine, et que le modèle multi-agents est lui-même inspiré de systèmes naturels ou sociaux.

1.3.4 Langages de communication entre agents

Les premiers langages de communication entre agents (ACL pour Agent Communication Language) ont ensuite fait leur apparition. Les deux plus répandus sont KQML [14], de Finin, Labrou et al. [140], [126] et FIPA ACL [44] de la FIPA [15]. Ces langages de communication ont été directement inspirés de la théorie des actes de langage, avec une certaine connotation " connaissance " héritée de l'IAD : les agents sont parfois perçus comme des bases de connaissances indépendantes. Ensuite, certains chercheurs ont associé à ces langages une sémantique formelle (voir [139]) afin de pouvoir raisonner sur les actes de langages, et surtout de supprimer certaines ambiguïtés d'interprétation.

Bien que les auteurs s'attachent à marquer une certaine différence avec l'intelligence artificielle conventionnelle, ces travaux adoptent une vision très rationnelle de la communication, et se sont tournés vers le raisonnement symbolique et la manipulation de connaissances.

La nouveauté est que les connaissances sont réparties, et que de multiples points de vues autonomes coexistent. Le problème est alors la confrontation et surtout la coopération entre ces multiples points de vue subjectifs. La standardisation de ces langages de communication entre agents est principalement confinée au niveau communicationnel, ce qui constitue une clef pour l'interopérabilité. Leur sémantique concerne avant tout l'interaction.

1.3.5 Protocoles d'interaction

L'interaction entre les agents est plus qu'un simple échange de messages [78] : un des aspects d'interaction montre une conversation basée sur un échange partagé et conventionné de messages. Les communications entre les différents agents dans les SMA sont souvent structurées selon des schémas typiques appelés protocoles d'interaction [119].

Un protocole d'interaction est un ensemble de règles et de contraintes communicationnelles, associées à un ensemble fini de rôles qui seront joués par des agents, les participants, au cours d'une conversation vérifiant ce protocole. Il est important de souligner qu'un rôle est intimement lié avec le protocole d'interaction dont il fait partie. La spécification d'un rôle fait nécessairement référence aux autres rôles, avec lesquels il est censé interagir, même si elle ne spécifie pas complètement leurs comportements.

14. Knowledge Query and Manipulation Language
15. Foundation for Intelligent Physical Agents

1.3.6 Apport des protocoles d'interaction dans la conception de l'intégration des processus métiers

L'intégration d'applications pas les processus partage avec les SMA plusieurs propriétés dont les méthodologies basées sur des agents tiennent explicitement compte dans leur ébauche de solutions. De telles propriétés sont [36][29] :

La répartition géographique et/ou logique d'entités, de données ou d'informations hétérogènes ;

La complexité du problème global à résoudre, celui-ci n'étant manipulable que par des stratégies heuristiques utilisant des données ou des connaissances locales ;

La flexibilité des interactions : il n'y a pas d'affectation a priori des tâches et les mécanismes de résolution de problèmes ne sont pas préalablement assignés à telle ou telle autre entité ;

L'environnement dynamique nécessitant, pour la résolution de problèmes, des entités réactives et adaptatives ;

L'ouverture : il n'est pas possible de donner une spécification complète du problème à résoudre ni de définir une fonction d'utilité globale.

En conséquence, les protocoles d'interaction des SMA semblent alors être particulièrement bien adaptés à la modélisation de la problématique d'intégration des processus métiers (de répartition, d'ouverture, de flexibilité et de collaboration des processus). C'est la raison pour laquelle l'objet de notre étude va porter sur l'étude des formalismes des protocoles d'interaction empruntés aux SMA pour la modélisation des problématiques de l'intégration. Plus précisément, notre travail a pour but de définir une approche d'ingénierie des protocoles, supportant des interactions flexibles des agents pour la satisfaction de besoins d'intégration énoncés par les entreprises, dans les contextes ouverts et décentralisés du Web. Notre objectif est alors de spécifier, vérifier, exploiter les protocoles d'interaction SMA et de les adapter pour le contexte d'intégration d'applications par les processus.

1.4 Formalismes de représentation des protocoles d'interaction

Toute modélisation passe par une étape de spécification, basée sur un modèle, et utilise un formalisme de représentation. La tendance actuelle veut que les formalismes graphiques, sous forme de diagrammes, aient plus de succès que les représentations purement textuelles. Cette prévalence se justifie dans notre cas par le fait que les protocoles d'interaction

constituent une perspective relativement abstraite, à l'échelle d'un agent, ce qui les rend propices à être décrits par des formalismes graphiques. Bien entendu un formalisme textuel peut toujours y être associé.

Cette section présente un état de l'art critique des formalismes et des représentations les plus couramment utilisés à cette fin. Nous verrons en particulier dans quelle mesure ces formalismes permettent de représenter les besoins de l'intégration des processus introduits précédemment.

1.4.1 Formalismes semi-formels

Afin de gérer la variabilité du nombre d'agents dans le protocole d'interaction, les langages graphiques considèrent non plus seulement les agents mais leur rôle dans le cadre d'interaction.

Les formalismes graphiques les plus utilisés dans le domaine SMA sont le graphe de Dooley et AUML, que nous présenterons d'une manière détaillée dans ce qui suit.

Graphe de Dooley

Le graphe de Dooley [102] est un formalisme graphique qui permet d'analyser *les actes de langage* dans le domaine des SMA. Selon Van Parunak [102], chaque agent a plusieurs états mentaux ou états de connaissance au cours de la communication et des sous-problèmes sont visualisés au cours de la résolution du problème. Un graphe de Dooley se compose des éléments suivants (voir fig. 1.4) :

- Nœuds : un nœud est un participant dans un certain état. Un trait en pointillé relie entre les états successifs d'un même participant. Par exemple $A_1, A_2, \ldots, A_i$, représentent des états de connaissances différentes.
- Arcs : un arc est un message dans la conversation (soit, un acte de langage qui reprend la sémantique utilisée en réalité : proposition, réponse.).

Dans ce graphe, quatre types de relation sont possibles pour décrire la conversation : "Respond", "Reply", "Resolve", "Complete".

- Respond = réponse de l'interlocuteur à un message,
- Reply = réponse au problème posé, en passant par des détours ; une réponse indirecte est un Reply,
- Resolve = introduit de la sémantique,
- Complete = met fin à l'échange.

Dans ce formalisme, le premier message n'a jamais de nom. Il introduit seulement une relation. L'ensemble des états et des messages définit des sous-ensembles identifiés de résolution des problèmes. Il existe des messages non classés dans le graphe, car ils ne correspondent pas à des

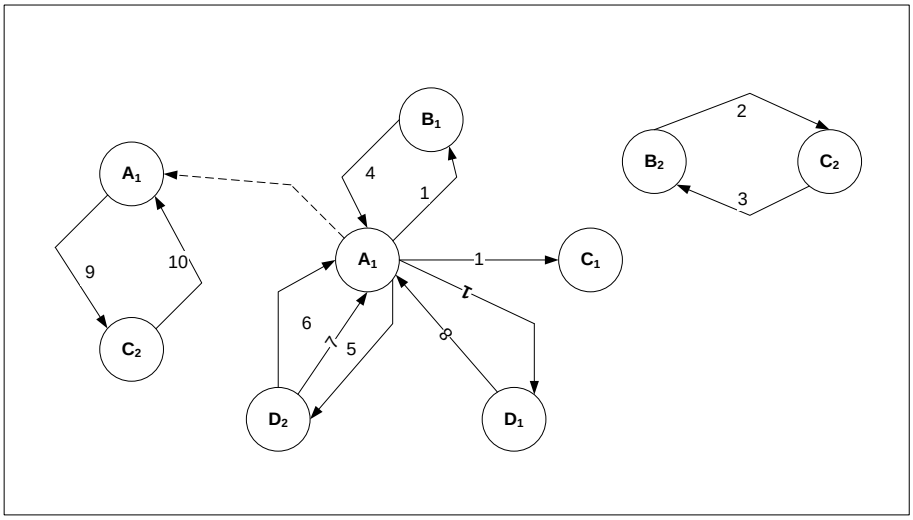

FIGURE 1.4 – *Représentation des protocoles d'interaction avec le graphe de Dooley [102]*

relations entre acteurs. Cela peut être une nouvelle information introduite, non classée dans le graphe et qui introduit en fait un nouveau scénario.

Les avantages de cette représentation se résument aux points suivants :

- Les participants de la conversation ainsi que les messages échangés sont représentés en même temps,
- Les nœuds représentent des états mentaux des participants durant les échanges.

Le problème majeur constaté dans ce formalisme est l'apparition de sous graphes non connectés lorsque :

- Des sous-conversations pour la résolution de sous problèmes,
- Des problèmes de communication au sein de l'organisation (croyances fausses sur les engagements, les connaissances d'autrui, etc.).

Notation AUML

AUML [10] est un langage de modélisation orienté agents. Il a pour but d'étendre UML [50] avec des moyens adaptés aux SMA. Ces derniers ont été souvent caractérisés comme une extension des systèmes orientés objets. Agent-UML a comme but de recommander une technologie pour l'adoption d'une sémantique, d'un méta-modèle et d'une syntaxe abstraite standard pour l'analyse et la conception des méthodologies basées sur les agents. Cette technologie devra couvrir le cycle de vie des produits et des outils de travail AUML et être en accord avec les spécifications existantes de FIPA (Foundation for Intelligent Physical Agents) et OMG [16].

Les notations proposées sont :

16. Object Management Group

- Une représentation des processus simultanés d'interaction permettant ainsi à UML de modéliser les protocoles d'interactions entre agents. (Par exemple le cas de transmission de messages à plusieurs destinataires)
- La notion de rôle qui permet de modéliser les rôles joués par les agents.

AUML a introduit le diagramme de protocoles qui est une extension du diagramme de séquences d'UML mais avec des opérateurs logiques, de causalité, de synchronisation et de diffusion [11]. La présentation sous forme de diagramme de séquences apporte une certaine intuitivité. La représentation explicite des rôles, et les éléments supplémentaires permettent la représentation des branches alternatives et du parallélisme. Aussi, AUML permet l'utilisation des boucles de retour ainsi que la définition de sous-protocoles. La figure 1.5 montre un diagramme AUML représentant le protocole Contract-Net [46].

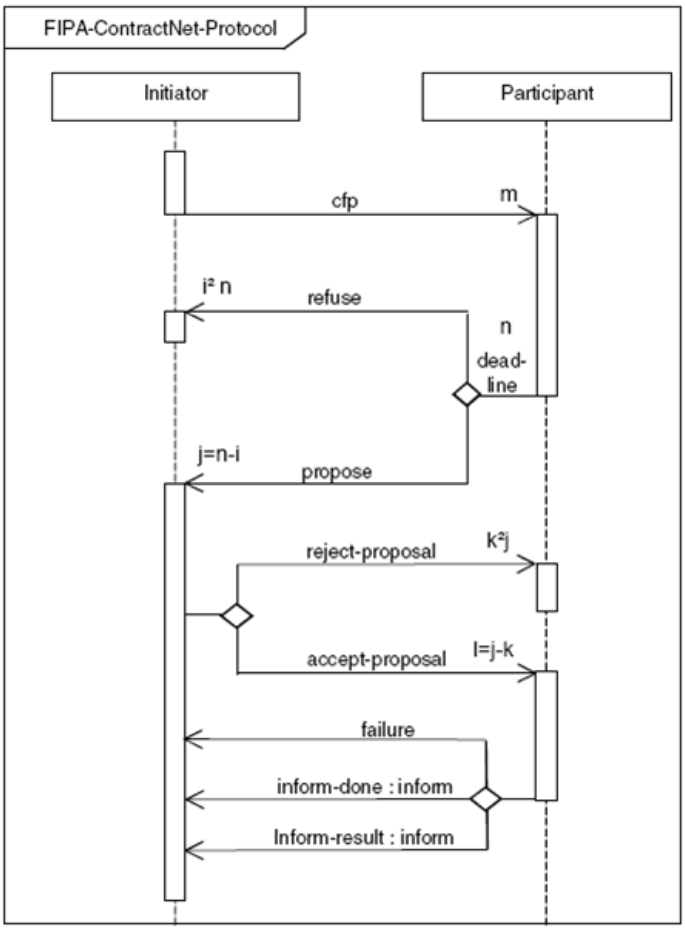

FIGURE 1.5 – *Représentation du protocole Contract-Net avec AUML [46]*

Les branches alternatives et parallèles (figure 1.6) sont introduites par des opérateurs graphiques (carrés en biais), appliqués soit aux lignes de

vie, soit aux envois de messages : une barre correspond à un *et logique parallle* (figure 1.6-a) ; un carré vide signifie un *ou logique* avec parallélisme potentiel (figure 1.6-b), et un carré muni d'une croix indique une alternative(figure 1.6-c).

Un raccourci d'écriture permet de représenter l'arrivée des messages de différentes alternatives sur une seule et même ligne de vie. La sémantique est alors équivalente à la réception des messages sur des branches parallèles, séparées en plusieurs lignes de vies.

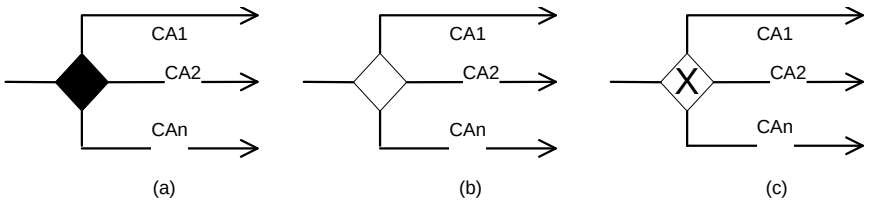

FIGURE 1.6 – *Connecteurs en AUML*

Synthèse sur les formalismes semi-formels

Nous avons vu dans la section précédente deux formalismes graphiques d'expression des PI. Cette section présente une comparaison de ces formalismes. Le choix des formalismes adéquats qui satisfait les besoins de représentation des PI, est guidé par quelques grands critères généraux. En fait, ces formalismes doivent :

- avoir un pouvoir d'expression explicite des différents participants dans le PI.
- permettre une représentation des alternatives et de parallélisme et
- permettre d'exprimer des contraintes sur l'envoi de messages

La représentation explicite des acteurs de protocole et de leurs interventions représente le principe essentiel de l'activité coopérative. Ainsi, cette représentation constitue une approche pour abstraire le fonctionnement ou le comportement d'un groupe d'agents en terme de messages échangés. A ce stade, les formalismes de Dooley et AUML sont intéressants car ils combinent à la fois une représentation explicite des rôles, et de leur dynamicité. En effet, ces deux formalismes spécifient les rôles source ou destinataire en les préfixant aux performatifs des messages. En revanche, dans le diagramme statecharts d'UML, les rôles en tant qu'entités adressables n'étaient pas représentés explicitement par un symbole graphique ou autre.

Dans le formalisme AUML, la représentation explicite des rôles (et des messages échangés) est plus intuitive que celle de graphe de Dooley. Ce dernier ne dispose pas du connecteur qui permet la représentation des alternatives, ce qui rend le graphe illisible, surtout dans le cas des sous-

conversations pour la résolution de sous problèmes (apparition de sous graphes non connectés).

L'utilisation des éléments supplémentaires pour la représentation des branches alternatives et du parallélisme (le cas de AUML) permet de distinguer plus facilement les différents chemins dans le protocole et de pouvoir les suivre plus facilement. Si l'on désire connaître tous les chemins qui partent d'une alternative, il suffit de prendre le contenu de l'élément graphique associé (qui soit un OU, XOR ou AND) à celui-ci. Dans le graphe de Dooley, il est nécessaire de suivre tous les nœuds représentent les états mentaux des participants durant les échanges

Un autre avantage de AUML est le fait que ce formalisme permet une représentation claire des contraintes sur le même diagramme. A ce stade, l'envoi de message peut se faire de plusieurs cas différents (voir figure 1.7). Chaque cas correspond à :

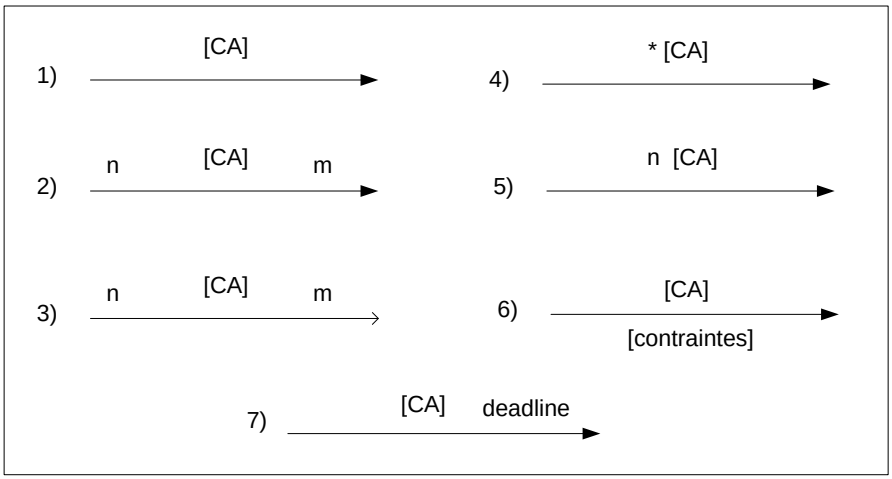

FIGURE 1.7 – *Différentes flèches pour l'envoi d'un message*

- Un envoi d'un rôle d'agent à un autre sans aucune contrainte (cas 1)
- Une réception de p messages avec n < p < m. Tous les messages doivent être reçus avant d'être traités. Il s'agit d'une synchronisation de messages (cas 2).
- Une réception de p messages avec n < p < m. Les messages sont traités au fur et à mesure (cas 3).
- Un envoi de message avec un nombre arbitraire de fois. Si l'astérisque est remplacé par un signe plus, le message est envoyé au moins une fois (cas 4).
- Un envoi de message *n* fois. Le nombre n peut être remplacé par un intervalle [n,m]. Dans ce cas, le message est envoyé p fois avec n < p < m (cas 5).
- Le message est envoyé uniquement si les contraintes sont vérifiées (cas 6).

– Le message doit être reçu avant que l'échéance ne soit dépassée (cas 7).

1.4.2 Modèles et langages formels

Nous présentons dans cette section quelques modèles et langages formels pour représenter les protocoles d'interaction. Un certain nombre de formalismes existent à l'heure actuelle pour la description d'un protocole tels que les automates à états finis, les réseaux de Petri, la logique temporelle et les langages Z[122], LOTOS [125], Estelle [73] ou SDL[65]. Cette section se concentre sur les trois premiers formalismes de description des protocoles d'interaction.

Les automates à états finis

Les automates à états finis ont été largement utilisées pour modéliser les protocoles d'interaction. Ceci a motivé l'étude des approches formelles de test des automates à états finis, afin de découvrir les aspects de leurs comportements et de s'assurer de la validité de leur fonctionnement. Le principe d'un automate à états fini est qu'il s'agit d'une succession d'états. Le changement d'un état à un autre correspond à l'envoi ou la réception d'un message.

Formellement, un automate à états fini est un quadruplet [67] : <Q, q_0, A, T> où :

Q : ensemble fini des états

q_0 : état initial de l'automate

A : alphabet de messages

T : fonction de transition d'états : QxA $\rightarrow Q$

L'intérêt des automates à états finis est, d'une part, de pouvoir décrire très facilement le comportement d'un agent capable de mémoriser l'état dans lequel il se trouve et, d'autre part, d'être soutenus à la fois par un support théorique important et d'avoir été utilisés dans de nombreux domaines de l'informatique. Plusieurs travaux sur la modélisation des protocoles sont basés sur ce formalisme [98], [58]. Cependant, ils s'avèrent très rapidement limités pour représenter des comportements plus complexes, et encore moins pour décrire l'évolution des interactions essentiellement pour deux raisons :

– Leur capacité de calcul est limitée, du fait que leur nombre d'états est fini.

– Ils ne peuvent décrire que des processus séquentiels. De ce fait, toutes les représentations parallèles sont pratiquement interdites. Il faut passer alors à des formalismes plus puissants et adaptés à la concurrence tels que les réseaux de Petri.

Les réseaux de Petri

Les réseaux de Petri (RdP) sont un formalisme de représentation de systèmes concurrents introduit par Carl Adam Petri [103]. Depuis lors, ce modèle formel a été très largement analysé et utilisé pour représenter des systèmes concurrents communiquant de façon asynchrone.

Formellement, un RdP est un quadruplet $RdP = (P, T, Entrée, Sortie)$ où :

- P est un ensemble fini de places
- T est un ensemble fini de transitions
- Entrée est une application, Entrée : $PxT \rightarrow N$, appelée application d'incidence avant
- Sortie est une application, Sortie : $PxT \rightarrow N$, appelée application d'incidence arrière

On appelle réseau de Petri marqué $R =< Q, M_0 >$ où M_0 est le marquage initial.

Par comparaison avec un diagramme à états finis standard, un réseau de Petri ne comprend pas d'états, mais des places, dans lesquelles circulent des jetons. L'originalité est que ces jetons circulent en parallèle, et une place peut donc à un instant donné contenir plusieurs jetons. Pour changer de place, un jeton doit déclencher une transition, qui est l'élément de synchronisation. Les places sont reliées à des transitions par des arcs orientés. Chaque transition ne peut être exécutée que si tous les arcs entrants apportent un jeton.

Les RdP sont génériques et ont été dérivés en un certain nombre de types de réseaux de Petri comme les réseaux colorés[14], récursifs[57], temporels[90], temporisés[112] ou à objet[42]. Seul le premier type de réseaux de Petri est introduit car il intervient dans la suite du manuscrit.

Les réseaux de Petri colorés

Les réseaux de Petri colorés (RdPC) [14] sont, comme tous les réseaux de Petri, composés d'un ensemble de modules qui contiennent chacun un réseau de places, de transitions et d'arcs. La spécificité des réseaux colorés est d'associer une couleur de marque différente à chaque processus. Les réseaux de Petri colorés sont, par exemple, utilisés dans les domaines des protocoles de communication, les systèmes répartis ou encore les systèmes de production automatisés.

Un RdPC est un sextuplet $RdPC =< P, T, C, C_{sec}, Pre, Post >$ où :

- P = $P_1, ..., P_n$ est un ensemble fini non vide de places ;
- T = $T_1, ..., T_m$ est un ensemble fini non vide de transitions ;
- C = $c_1, ..., c_r$ est un ensemble fini non vide de couleurs ;
- $C_{sec} : P \cup T \rightarrow P(C)$ est la fonction sous-ensemble de couleurs qui à

chaque place et à chaque transition est associé un sous ensemble de C ;

- *Pre* est l'application définie sur PxT qui à tout couple (P_i, T_j) associent une application définie de $C_{sec}(T_j)$ dans l'ensemble des applications de $C_{sec}(Pi) \rightarrow N$.
- *Post* est l'application définie sur PxT qui à tout couple (P_i, T_j) associent une application définie de $C_{sec}(T_j)$ dans l'ensemble des applications de $C_{sec}(P_i) \rightarrow N$.

Les RdPC ont été largement utilisées pour modéliser les protocoles d'interaction. Lin et al. [76] définissent une méthode de spécification de protocole en RdPC à partir d'une description exhaustive des états et transitions associées à des échanges de messages. Les RdPC produits sont plus faciles à lire, car ils utilisent peu de coloration et des conditions de garde. Cependant, cette formalisation aboutit à une représentation plus verbeuse en termes de places et de transitions. Chaque transmission de message correspond à une place, en plus des nombreuses places supplémentaires peuvent êtres ajoutées pour représenter les états de chaque rôle. Les auteurs proposent une validation du protocole à l'aide d'outils de *simulation* de RdPC.

Cost et al. dans [111] ont porté leur choix sur RdPC, bien qu'initialement la plateforme Jackal, de la même équipe, utilisait une machine à états finis étendue pour gérer les conversations. Les principales motivations de ce choix reposent sur la possibilité de représenter la concurrence, et sur l'existence d'outils de simulation et de validation. Ils définissent une stratégie de transcription pour passer d'une représentation à l'autre. Cette transcription dédie une place $input$ pour l'arrivée des messages de chaque agent, et donc nécessite d'utiliser la coloration des arcs pour distribuer les messages entrants.

Elfallah-Segrouchni, Haddad et Mazouzi dans [55][2] proposent également les RdPC pour modéliser les protocoles d'interaction, et justifient leur choix de la même manière. Ils décrivent dans [2] une transcription des diagrammes AUML en RdPC. La structure des RdPC résultants est proche de celle de Lin et al [76].

Un inconvénient de ces approches est que le passage des différents modèles des PI vers le modèle de RdPC nécessite un grand effort. En effet, aucun outil automatique a été développé pour la génération de RdPC à partir des spécifications des PI.

Logique temporelle

Les logiques temporelles permettent de représenter et de raisonner sur certaines propriétés de sûreté des systèmes. En ce sens, elles sont

bien adaptées à la spécification et à la vérification des systèmes réactifs et concurrents [104]. Elles pourraient donc être adaptées à la spécification des protocoles d'interaction. Néanmoins, il ne faut pas oublier que ces logiques ont été conçues pour raisonner sur des propriétés usuelles des systèmes concurrents comme l'invariance ("de mauvaises choses ne peuvent pas se produire"), la vivacité ("les bonnes choses vont effectivement se produire"), ou encore la précédence ("il faut avoir préalablement demandé une ressource pour espérer l'obtenir").

On distingue deux grandes classes de logique temporelle selon la structure du temps considérée [20] :

1. logique temporelle linéaire (LTL)
2. logique temporelle arborescente (CTL)

Dans le cadre des logiques temporelles linéaires le temps se déroule, comme son nom l'indique, linéairement. En clair, on spécifie le comportement attendu d'un système sans réelle possibilité de spécifier plusieurs futurs possibles.

La temporalité de cette logique intervient par l'intermédiaire des opérateurs. Voici par exemple les opérateurs :

- Les connecteurs booléens : $\vee, \wedge, \neg, \Leftarrow, \Rightarrow, \Leftrightarrow$.
- Les quantifieurs du premier ordre : $\forall$ et $\exists$
- Les opérateurs temporels : $\bigcirc$('next'=prochain), $\Box$('always'=toujours), $\Diamond$ ('eventually'= finalement) et $\cup$ ('until'=jusqu'à).

Dans le cas de la logique CTL, deux dimensions sont à prendre en considération sur les ensembles de traces : l'aspect non déterministe des choix des branches en chaque état (issus des désynchronisations), et l'aspect linéaire d'une trace (i.e. les propriétés concernant la succession des états le long d'un chemin).

- Sur la première dimension, à partir d'un état initial donné, on peut demander qu'une propriété soit vraie pour au moins un choix de branche (" there Exists a path such that . . . ") ou bien qu'elle soit vraie pour tous les choix, c'est-à-dire tous les futurs possibles (" for All possible pathes . . . ").
- Sur la seconde dimension, partant d'un état initial le long d'un chemin donné, on peut demander qu'une propriété soit vraie : à l'instant juste après l'état initial (" neXt state "), au moins à un instant donné sur le chemin (" some Future state "), tout le temps (" Globally ") ou jusqu'à ce qu'une autre propriété devienne vraie (" . . . Until . . . ").

On obtient les modalités temporelles de CTL en accolant ces deux dimensions. Par exemple :

- $AX(\Phi)$ signifie que la formule Φ est vraie pour tous les états qui suivent immédiatement l'état initial ;
- $EF(\Phi)$ signifie qu'il existe une exécution (opérateur E) conduisant à un état où Φ est vérifié (opérateur F)».
- $EG(\Phi)$ signifie qu'il existe au moins un chemin partant de l'état initial le long duquel Φ reste tout le temps vraie ;
- $AF(\Phi)$: pour toute exécution , il existe un état où Φ est vérifiée ;
- $AG(\Phi)$: pour toute exécution, Φ est toujours vérifiée.
- $E\Phi U\Psi$ signifie qu'il existe une exécution (opérateur E) durant laquelle Φ est vérifiée jusqu'à ce que Ψ le soit ($\Phi U\Psi$).
- $A\Phi U\Psi$ signifie que pour toute exécution (opérateur A) Φ est vérifiée jusqu'à ce que Ψ le soit ($\Phi U\Psi$).
- $EX(\Phi)$: il existe une exécution (opérateur E) dont le prochain état satisfait Φ.
- $AG\ EF\Phi$: A partir de n'importe quel état atteignable, il est possible d'atteindre un état satisfaisant Φ.

L'intérêt de cette approche est de pouvoir axiomatiser le comportement et ensuite de définir les propriétés à vérifier à l'aide de formules en logique temporelle. La spécification en logique temporelle des comportements d'agents a fait l'objet de plusieurs travaux dont Wooldridge et al. [137], Dignum et al. [33] et dans Fisher et al. [45] pour la description du protocole Contract-Net.

1.4.3 Relation avec le travail de thèse

Au vu des différentes approches énoncées précédemment, il existe un réel attrait à la modélisation et la vérification de certaines propriétés sur l'intégration et la coordination des processus métiers. L'objectif de ce travail de thèse est de proposer une approche conceptuelle et technique orientée interaction pour l'intégration des processus métiers.

La première étape de notre travail consiste à définir de manière graphique (ou semi-formelle), le protocole d'intercation. Pour cela, le formalisme AUML est utilisé afin de capturer l'échange de messages entre les différents processus métiers.

Le diagramme d'interaction AUML est très lié au problème d'intégration des processus métiers. Il permet de définir un modèle abstrait d'interaction entre les différents collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires.

Une description graphique n'est pas facilement manipulable par des outils, il est donc important qu'il existe une version textuelle de cette représentation graphique. Elle est fournie dans une deuxième étape de notre

travail. En fait, cette étape permet d'enrichir la modélisation AUML avec le langage BPEL4WS qui constitue un standard lié aux services Web.

D'une façon générale, le langage BPEL4WS décrit les relations entre les services Web (les services Web composés). Le choix de ce langage a été influencé pour divers raisons :

- Le langage BPEL4WS offre une représentation textuelle et détaillée des PI. Cela permet aux agents d'exploiter facilement les PI. En faite, les agents qui représentent les processus métiers locaux doivent a priori savoir les relations entre eux. Ces relations sont intégrées dans le fichier ProcessLogic du BPEL4WS.
- Aussi, le langage BPEL4WS permet facilement la publication de ces PI sur le Web (pour des raisons de réutilisation).
- Notons enfin qu'il existe une forte corrélation entre les deux formalismes (le concept d'interaction est à la base de la définition de BPEL4WS).

En outre, notre approche consiste à proposer une sémantique formelle des protocoles d'interaction à l'aide des RdPC. Cette formalisation nous permet d'appliquer des outils pour la vérification et la validation. A partir de là, nous pouvons valider le protocole à l'aide de l'outil CPN-AMI [17]. Nous utilisons pour cela la logique temporelle pour exprimer l'ensemble des propriétés que nous souhaiterons vérifier.

1.5 Conclusion

Ce chapitre a permis d'effectuer une étude des différents concepts que nous utiliserons dans le reste de cette thèse. En effet, nous avons vu dans la première partie de ce chapitre que plusieurs approches se sont déjà penchées sur le problème de l'intégration de processus métiers. Certaines utilisent des techniques de workflow et de serveurs d'application, des middleware orientés messages, ou des mécanismes basés sur les services Web.

Dans la deuxième partie de ce chapitre, nous avons étudié les différents modèles et langages liés à la formalisation des protocoles d'interaction.

Après une analyse de ces différents travaux, il en ressort que la difficulté pour l'intégration d'applications basée sur la coordination des processus réside principalement dans :

- La formalisation et la vérification des scénarios d'intégration (l'interaction entre les différents partenaires). La plupart des approches existantes ne traitent pas ce problème puisqu'elles s'appuient sur la notion de workflow, écrit par un expert dans un langage tel que WS-BPEL, destiné à être instancié et exécuté d'une part, et manipulé par des spécialistes en technologie de l'information d'autre part.

17. http ://www.lip6.fr/cpn-ami

– La conception d'un modèle dynamique d'interaction des processus métiers, qui tient compte à la fois le but de l'intégration et des fonctionnalités des différents partenaires.

Par ailleurs, notre étude nous a conduit à la conclusion que les protocoles d'interaction définis dans le domaine SMA étaient un bon candidat pour pallier les limites des techniques d'intégration d'applications par coordination des processus. Le concept de protocoles d'interaction apporte des solutions et des modèles pour tenir compte de plusieurs contraintes telles que la répartition des partenaires, l'ouverture et la flexibilité de leur environnement. C'est sur ces points que notre travail porte. En effet, les buts d'intégration, décrits à travers des protocoles d'interaction dans notre cas, spécifient le profil des processus à intégrer ainsi que la séquence de leur invocation. Ces protocoles sont exploités ensuite par le SMA afin d'interagir de manière flexible et de couvrir adéquatement le but d'intégration.

2 Une approche orientée Interaction pour la modélisation de l'intégration des processus métiers

Sommaire

Dans ce chapitre nous nous intéressons à l'intégration des processus métiers dont l'interaction se base sur les protocoles d'interaction. Le développement de ces systèmes nécessite une bonne maîtrise des protocoles d'interaction et de tous les détails de leur utilisation. Le développeur doit connaître le langage de communication à utiliser, tels que FIPA-ACL, l'enchaînement des messages et la spécification de leur contenu. De ce

fait, l'implémentation des protocoles d'interaction est complexe, de même pour l'implémentation des agents qui les utilisent.

Pour palier ce problème, ce chapitre a pour but de proposer une «approche orientée Interaction», permettant la modélisation de l'intégration des processus métiers. Le concept des protocoles d'interaction nous a permet de définir les spécifications du protocole, i.e. l'ensemble des fonctionnalités et des propriétés qui doivent être présentes dans le scénario d'intégration des processus métiers.

A ce stade, nous avons utilisé le formalisme graphique AUML[10] accompagné d'un autre formalisme textuel BPEL4WS[64] pour la modélisation des actions de communication entre les différents processus métiers.

Nous proposons par la suite, une sémantique opérationnelle de notre modèle. Cette sémantique est basée sur des règles de transformation garantissant le respect d'une certaine cohérence du système. Le modèle formel retenu pour la représentation du comportement observable des processus métiers est les réseaux de Petri colorés (RdPC) [14].

L'aspect formel nous permet de représenter d'une façon précise les interactions des processus, de vérifier certaines propriétés inhérentes à cette interaction telle que l'interblocage et de valider les exigences décrites dans le cahier des charges lors de la phase d'analyse.

2.1 Travaux antérieurs

Plusieurs travaux de recherche ont tenté de rapprocher la modélisation des SMA avec celle des systèmes d'information, dans différentes optiques [30].

Certains de ces travaux ont proposé des modèles pour représenter les SMA, dont Kishore et al. [108] ont effectué une large revue. Dans cette perspective, les seuls acteurs sont des agents intelligents, et les acteurs humains sont généralement considérés comme des utilisateurs du système. Certains auteurs ont utilisé des langages de modélisation issus du génie logiciel, notamment UML [63].

D'autres travaux ont abordé la problématique de la modélisation des communications entre organisations, en utilisant notamment le concept de contrat [56], [138]. D'autres encore ont proposé une vision d'un processus métier comme une succession de cycles de communication entre les différents participants [52]. Kishore et al. [108], proposent un cadre unificateur entre la modélisation des SMA et celle des systèmes de coordination, permettant de spécifier les interactions entre processus métiers.

Wagner [134] propose un métamodèle basé sur la distinction entre les agents, les éléments actifs, les objets et les éléments passifs. Le métamodèle proposé est orienté vers la représentation des communications.

El Fallah-Seghrouchni et Haddad [55][2] ont proposé un modèle récursif pour la représentation et la manipulation des plans en utilisant les réseaux de Petri récursifs. Ce modèle récursif présente des propriétés qui sont similaires au framework que nous proposons. Ces propriétés consistent à supporter les activités concurrentes, la réaction à l'environnement en utilisant la dynamicité d'exécution de plans, les actions abstraites et le raffinement dynamique, et la résolution d'interaction en utilisant la coordination de plans. Cependant, le plan global est supposé non nécessaire puisque l'approche considérée est basée sur le paradigme de planification orientée agent. En fait, cette supposition ne convient pas aux applications de BPI où la planification d'un comportement global est nécessaire.

Marc et al [39] ont proposé une approche dont le comportement global du système est modélisé à un certain niveau d'abstraction. Les auteurs montrent qu'il est possible de modéliser les plans des agents en termes d'automates hybrides. Le SMA associe à chaque tâche réalisée un graphe de dépendances qui représente la décomposition d'une tâche en sous-tâches. Les principaux avantages de ce formalisme sont le contrôle et la validation des plans individuels et multi-agents.

Buhler et al. [7][99] ont proposé l'utilisation des services Web et des SMA pour assurer l'interopérabilité des workflows. Les auteurs résument

le lien entres les deux technologies par l'aphorisme : *"Adaptive Workflow engines = web services + agents"*. A ce stade, les services Web fournissent les ressources de calcul et les agents fournissent le cadre de coordination des workflows. L'approche préconisée utilise BPEL4WS [64] comme langage de spécification pour exprimer les interactions de système multi-agent. Un inconvénient de cette approche est que le passage des différents modèles vers le modèle agent nécessite un grand effort. Par ailleurs, aucune étape de validation afin d'assurer l'exactitude des interactions entre les agents n'est prévoie.

D'autres travaux reliés à la communication entre agents et la modélisation des systèmes ont procédé du point de vue du protocoles d'interaction. Comme l'utilisation d'un langage naturel est illimitée, la théorie des actes de langage [121] [84] exprime qu'il est possible de grouper les termes en un nombre fini de groupes d'actions verbales baptisés *"les actes de langage"*. Cette théorie introduit la notion de protocole de communication pour représenter les différents comportements des agents sociaux. Comme il a été détaillé dans [115], la théorie des actes de langage est utilisée dans plusieurs applications dans les domaines des systèmes répartis, des protocoles d'échange de données électroniques, etc. L'approche utilisée dans l'IAD exploite cette théorie pour formaliser les protocoles de communication entre les agents. Singh propose dans [86] une classification en sept catégories d'actes : assertions, expressions, déclarations, promissions, permissions, directions et prohibitions.

Medina-Mora et Denning [109] [32] utilisent les actes de langage pour modéliser les processus centrés sur la coopération comme un workflow à quatre phases cycliques : demande, négociation, performance et satisfaction. Chang et al ont proposé le protocole SANP (Speech Act-based Negotiation Protocol) [79] dont la conception est basée sur l'intégration de deux domaines sociaux différents : la théorie des actes de langage en linguistique et la négociation en sociologie. La théorie des actes de langage utilise l'information basée sur l'ordre partiel laquelle n'est pas suffisante pour construire un protocole complet de négociation. Ceci est dû au fait que l'ordre partiel ne fournit pas l'information de type branchement ou de type raisonnement concernant un argument défié.

Desai et al. [93] proposent une ontologie générique pour la description des protocoles pour les processus d'entreprise. Dans leur description, ils utilisent des règles décrites dans le langage SWRL [62]. Leur ontologie regroupe des concepts importants tels que les messages, les rôles, les règles, et la notion d'engagement (commitment). Cette ontologie permet aussi la composition de protocoles. L'inconvénient de cette proposition est la faiblesse du pouvoir exprimer les protocoles en terme de structures de contrôle car elle ne permet pas de spécifier des protocoles complexes

(notamment concurrents et itératifs) mais offre seulement la possibilité de définir des échanges de messages assez simples. En plus, les auteurs n'envisagent aucune étape de validation afin d'assurer l'exactitude des protocoles.

Synthèse

A l'image de cette étude, nous relevons l'importance de l'approche agent et plus précisément, les protocoles d'interaction multi-agents pour modéliser les interactions entre processus métiers.

En plus des critiques apportées séparément à chacune des approches décrites ci-dessus, nous avons pu constater l'absence d'une démarche complète pour l'ingénierie des protocoles d'interaction dans le cadre d'intégration d'applications.

La plupart des travaux mentionnés dans la section précédente, se concentrent sur une étape précise. Par exemple, le cadre conceptuel proposé par Kishore et al. [108] traite la modélisation des processus métiers à l'aide des SMA. Dans ce cas, les auteurs ne prévoient aucune étape de validation afin d'assurer l'exactitude des modèles développés.

Les travaux de El Fallah-Seghrouchni et Haddad [55][2] traitent seulement les étapes de modélisation et de vérification à l'aide des réseaux de Petri récursifs. L'exploitation (au moment de l'exécution) des modèles d'agents développés est complètement ignorée.

Dans le cadre de cette thèse, notre travail comporte deux contributions qui couvrent les principales étapes de modélisation et de réalisation :

- La modélisation et la vérification du scénario d'intégration (objectif de ce chapitre).
- L'exploitation et la gestion de l'interaction au moment d'exécution (objectif de chapitre suivant).

Dans le reste de ce chapitre, nous proposons un cadre conceptuel et une démarche méthodologique couvrant les principales phases pour la modélisation de BPI basée sur les PI [24][25]. Cette démarche est basée conjointement sur l'utilisation de la notation Agent-UML (AUML) [11][10], et aussi sur le langage BPEL4WS [64].

Nous définissons par la suite un guide permettant de passer à partir de descriptions semi-formelles des PI, vers des spécifications formelles de réseaux de Petri colorés (RdPC) tout en mettant l'accent sur le contrôle et la dynamicité des interactions. Le modèle de RdPC permet de couvrir les aspects de modélisation et de validation des protocoles.

2.2 Vue globale de l'approche proposée

La collaboration au sein des systèmes d'entreprises est essentiellement gérée par les processus métiers. La notion de processus permet de modéliser la dynamique des informations et de définir de manière cohérente les comportements des différents objets manipulés [132] [80].

Dans notre travail, nous distinguons deux grandes catégories de processus : privés et publics. Les processus privés sont conçus pour satisfaire des besoins locaux et non nécessairement pour travailler ensemble.

Nous parlons de processus public lorsqu'une application ou un client requiert des fonctionnalités, et qu'aucun processus privé n'est seul apte à les fournir. Dans ce cas, il devrait être possible de combiner des processus existants afin de répondre aux besoins de cette application ou de ce client. C'est ce que l'on appelle l'intégration des processus métiers.

L'intégration d'applications dépeint les interactions dans lesquelles les processus engagés participent afin d'atteindre un objectif commun. Ces interactions représentent les dépendances entre les différents participants telles que le flot de contrôle (ex. un message doit précéder une autre), le flot de données, les corrélations des messages et les contraintes de temps, etc.

Dans cette perspective, l'interaction est un mécanisme important pour supporter le partage des ressources et coordonner les activités entre les différentes applications de l'entreprise. Le concept de protocole d'interaction est un moyen efficace pour structurer et organiser les échanges entre ces applications. Au delà des protocoles de bas niveau qui assurent la communication, l'intérêt se porte aujourd'hui sur des protocoles de haut niveau, issus des SMA. Ces protocoles ne se limitent pas à des échanges de type client-serveur mais exprimant des interactions complexes de type négociation, enchère, délégation et sous-traitance. Par ailleurs, dans le cadre des SMA, les protocoles se basent sur des langages à base de performatifs du type FIPA-ACL (propose, call for proposol, inform, subscribe,) qui enrichissent les échanges [43][15].

En conséquence, notre préoccupation dans ce travail de recherche concerne *l'intégration des applications par les processus* à l'aide d'agents dits communiquants, c'est à dire ils interagissent par l'envoi de messages, en utilisant un langage de communication de haut niveau. Plus précisément, l'approche que nous proposons se focalise sur les protocoles d'interaction qui constituent l'unité fondamentale du processus d'intégration.

Dans cette section, nous présentons un cadre conceptuel permettant l'intégration des processus métiers (voir figure 2.1). Cette intégration a pour objectif de permettre à plusieurs systèmes logiciels, de même na-

ture ou hétérogènes, de communiquer et de coopérer afin d'atteindre un objectif commun.

Notre approche d'intégration des processus utilise les protocoles d'interaction multi-agents se basant sur des agents offrant des services. En effet, les processus privés qui ont fait partie de la même entreprise sont représentés par un agent logiciel baptisé *agent d'entreprise*. Ce dernier sert d'interface entre l'entreprise et son environnement.

Le processus public est spécifié par les protocoles d'interaction. Ces derniers permettent de concevoir le scénario d'intégration en terme des messages échangés et un ensemble de contraintes métiers.

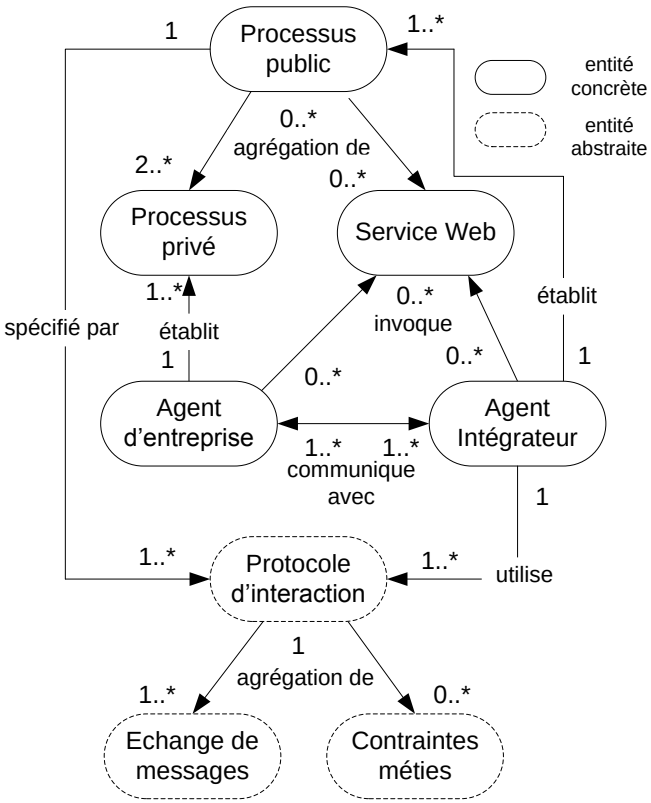

FIGURE 2.1 – *Cadre conceptuel de l'approche proposée*

Les protocoles d'interaction sont ensuite utilisés par un agent baptisé *Intégrateur* pour établir le processus public au moment d'exécution. Nous rappelons qu'un PI définit la forme et la séquence des messages à échanger par les agents. En effet, l'agent *Intégrateur* utilise ces informations afin de coordonner avec les autres agents d'entreprises. Les PI devraient être publiés et partagés par l'agent *Intégrateur* pour des éventuelles réutilisations.

Dans le reste de ce chapitre, nous présentons notre démarche méthodologique couvrant les principales phases pour la modélisation de BPI.

2.3 Les protocoles d'interaction : une abstraction du scénario d'intégration

De notre point de vue, une abstraction intéressante de BPI se définit par la description de ses échanges observables. En effet, nous proposons les Protocoles d'Interaction (PI) comme une abstraction et une référence sémantique pour décrire les BPI indépendamment de leurs langages de définition.

Un PI est un concept qui a été emprunté des SMA. Il est utilisé comme un moyen d'abstraction de comportement des agents. En effet, nous introduisons l'idée de conception orientée interaction pour l'intégration des processus métiers.

Ainsi, spécifier le comportement des différents agents du système, selon la perspective d'un objectif collectif donné, revient à spécifier les interactions sous forme de protocoles, de rôles, et éventuellement d'intentions. Un PI représente la collaboration et la coordination de plusieurs rôles compatibles. Chaque agent peut être impliqué dans une ou plusieurs de ces collaborations, à différents moments de son existence, en endossant différents rôles [87].

Avant d'exposer notre relation d'interaction, nous détaillons ses particularités à travers la définition suivante.

Définition 1 : Un Protocole d'Interaction est un quadruplet :
PI = <ID, R, M, f_M>, où :

- ID est l'identifiant de PI ;
- $R = r_1, r_2, \ldots, r_n$ ($n > 1$) est un ensemble fini de Rôles (Processus métier privé ou Service Web) ;
- M est un ensemble non vide des messages primitifs (et/ou) des messages complexes, où
  - Un message primitif (PM) correspond au message simple, il est défini comme suit :
    PM = <$Sender, Receiver, CA, Option$>, où :
    - Sender, Receiver $\in$ R
    - CA $\in$ Actes de Communication de FIPA ACL (cfp, inform, propose,...)
    - Option : contient des informations supplémentaires (massage asynchrone/synchrone , contraintes sur le message,...)
  - Un message complexe (CM) est composé à partir des messages simples (primitifs) par le moyen des opérateurs.
    CM = $PM_1 op PM_2 \ldots op PM_m$, où :
    - $m > 1$, op $\in$ {XOR, OR, AND}, and
    - $\forall$ i $\in$ [1,m[, $PM_i.Sender = PM_{i+1}.Sender$, $PM_i.Sender \in R$.

– f_M : est une relation de transition définit comme suit : $f_M \subseteq$ (RxR), où (RxR) est le produit cartésien $(r_1, r_2) \in$ (RxR) et $r_1, r_2 \in$ R

La formalisation du processus d'intégration avec les PI retranscrit la possibilité d'une interaction correcte et complète, d'un point de vue opérationnel, entre deux ou plusieurs entreprises communicant et utilisant des spécifications différentes. Dans le cadre de notre travail cette formalisation se matérialise en deux étapes (figure 2.2) :

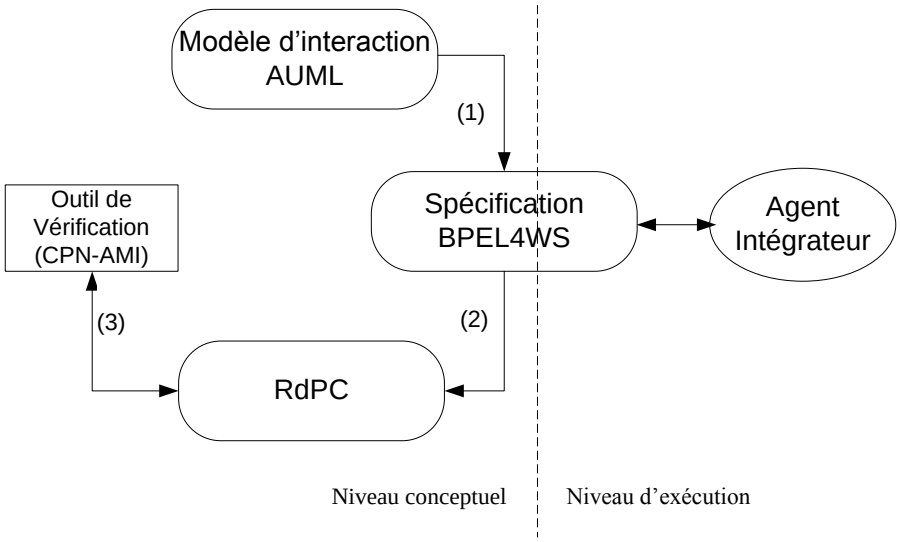

FIGURE 2.2 – *Approche proposée*

1. La définition d'un niveau d'abstraction commun afin d'exprimer la sémantique des différents participants dans l'interaction. Le niveau d'abstraction choisi dans ce travail se restreint aux actions de communication observables. En conséquence, le concept de PI semble être particulièrement bien adapté à la modélisation de l'intégration des processus métiers. Comme le montre la figure 2.2 nous avons proposé la combinaison AUML/BPEL4WS pour la spécification des protocoles.
2. L'utilisation d'un mécanisme formel pour exprimer la sémantique de notre modèle. En effet, nous définissons des règles de passage pour exprimer la sémantique des différentes constructions des langages de spécification de PI. Le modèle formel retenu pour la représentation du comportement observable des processus est le RdPC. L'aspect formel nous permet de représenter d'une façon précise les interactions des processus, de vérifier certaines propriétés inhérentes à cette interaction telle que l'interblocage et de valider les exigences attendues par le client.

2.4 Description informelle des protocoles d'interaction

Dans l'approche que nous proposons, le niveau d'abstraction choisi se restreint aux actions de communication observables entre les différents processus métiers. A ce stade, nous avons utilisé la notation AUML [10] pour la modélisations de ces actions de communication entre les différents systèmes (processus local ou service Web). Les diagrammes d'interaction d'AUML sont une extension des diagrammes de séquence UML existants qui permettent différentes cardinalités d'envois de messages (AND, OR et XOR).

Le modèle d'interaction d'AUML est très lié au problème d'intégration des processus métiers basés sur des services Web. Il permet de définir un modèle abstrait d'interaction entre collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires. Les processus métiers sont représentés par des rôles AUML, un flux d'événements sur lesquels on peut influer en définissant des règles métier, des règles de sécurité, des règles de transactions. Comme le montre le tableau 2.1, cette norme est à la base de la définition de BPEL4WS [64] qui constitue un standard lié aux services Web.

Cette forte corrélation nous a permis d'exploiter le langage BPEL4WS pour spécifier ces interactions (les actions observables décrites précédemment). Généralement le langage BPEL4WS décrit les relations entre les services Web (les services Web composés). Dans notre travail ce langage est utilisé pour deux raisons,

- Premièrement pour exploiter les PI à l'aide de SMA. Dans ce cas, les agents qui représentent les processus métiers locaux doivent a priori connaître les relations entre eux. Ces relations sont intégrées dans le fichier ProcessLogic du BPEL4WS.
- En plus, le BPEL4WS permet facilement la publication de ces PI sur le Web (pour des raisons de réutilisation).

2.4.1 Enrichissement des protocoles d'interaction avec le langage BPEL4WS

Comme nous l'avons souligné dans le chapitre précédant, un PI est une spécification qui guide la communication entre agents. Il précise qui peut dire quoi à qui et les réactions possibles à ce qui est dit par l'autre partie. La fonction de base d'un PI est de permettre aux agents de communiquer sans avoir besoin de planifier explicitement chaque acte du langage en se basant sur ses effets et actions attendus.

Dans cette section nous présentons des règles de traduction des diagrammes d'interaction AUML vers une spécification BPEL4WS.

AUML	BPEL4WS
Roles	<partners>
sequence	<sequence>
AND-split	<flow>
OR-split	<switch>
XOR-split	<if>
Iteration	<while>
Message Synchrone	<invoke>
Message Asynchrone	<receive>, <reply>

TABLE 2.1 – *Mapping AUML/BPEL4WS*

Dans notre modélisation, les actions observables lors de l'interaction des processus sont les échanges de messages. En conséquence, nous utilisons seulement les composants BPEL4WS qui nous permettent de modéliser l'interaction observée. Ces composants sont regroupés en plusieurs catégories :

- les composants de base : tels que *partner*, *throw* ;
- les composants basés sur les messages : tels que receive, reply, invoke ;
- les composants de contrôles structurés et évolués, : *switch*, if, *pick*, *while*, $flow$;

Le diagramme d'interaction d'AUML est traduit vers une spécification BPEL4WS nommée par le nom du protocole. Cette spécification inclut huit ensembles abstraits qui sont représentés dans la table 2.1. Dans le reste de cette section, nous abordons la traduction du squelette général des diagrammes d'interaction AUML vers des spécifications BPEL4WS.

Définition des partenaires de l'interaction

En AUML, les protocoles d'interaction sont présentés en général par des rôles. La partie Rôle de diagramme d'interaction est explicitement représentée par la section <*partners*> de BPEL4WS. Un partenaire est un processus (et/ou) service Web évoluant dans une interaction. Dans notre approche, chaque partenaire est représenté par une description BPEL4WS incluant :

- la description de ses partenaires (ou un lien permettant de l'obtenir) ;
- la description de son interface (et de ses opérations locales) ;
- la description du processus abstrait qu'il représente, définissant son comportement. Cette partie est généralement décrite dans le langage de description comportementale BPEL4WS.

Définition de l'interaction

L'interaction est une vue globale des actions observables des différents partenaires impliqués dans cette interaction. En fait, la spécification formelle de l'interaction est obtenue en associant chaque opération d'un partenaire à son partenaire cible.

Dans la suite de cette section, nous montrons les différentes adaptations nécessaires de la sémantique du langage BPEL4WS au vu de la modélisation des protocoles d'interaction. Dans notre contexte, l'accent est mis sur les composants basés sur les messages. Comme le montre la définition d'un PI que nous avons proposé dans la section 3.3, les messages que nous considérons sont de trois types : synchrone, asynchrone et complexe

Représentation des messages synchrones :

Dans la sémantique de BPEL4WS, le processus *invoke* ne définit aucun échange entre le service et le client : il se contente d'appeler une opération d'un autre service, à partir du service l'utilisant. Donc cela n'introduit aucun message supplémentaire dans l'interaction.

Dans notre cas, le processus *invoke* représente une communication synchrone où un expéditeur envoie une requête et attend la réponse.

Représentation des messages asynchrones :

Le processus *receive* est un processus BPEL4WS permettant au service Web de se mettre en attente de réception d'un message. Ce processus peut être suivi, dans certains cas, par le processus *reply* envoyant une réponse à l'émetteur du message reçu par *receive*.

En effet, les processus *invoke* et *reply* correspondent au mode de communication asynchrone dans laquelle l'exécution peut continuer sans avoir besoin de synchronisation. Dans ce cas et lorsque l'expéditeur envoie sa demande, il ne s'en occupe plus du tout (il oublie complètement qu'il a envoyé une demande) et il continue son exécution.

Représentation des messages complexes :

Comme le montre le tableau 2.1, les messages complexes de type AND-split, OR-split et XOR-split sont représentés respectivement par les processus BPEL4WS <$flow$>, <*switch*> et <if>.

Le processus $flow$ est utilisé afin de modéliser que plusieurs messages peuvent êtres envoyés en même temps. Alors que le processus *switch* permet de modéliser le "ou inclusif" où zéro, un ou plusieurs messages peuvent êtres envoyés en même temps. Le dernier processus BPEL4WS (le <if>) exprime qu'un et un seul message peut être émis à la fois. Le choix peut être effectué entre différents messages.

```
<invoke partnerLink="ncname"
     portType="qname"
     operation="ncname"
     inputVariable="ncname"?
     outputVariable="ncname"?>

  <catch faultName="qname"
     faultVariable="ncname"?>
     activity
  </catch>
  <catchAll>?
     activity
  </catchAll>
  <compensationHandler>?
     activity
  </compensationHandler>
</invoke>
```

FIGURE 2.3 – *Syntaxe BEPL4WS pour l'invocation d'un service Web*

Représentation des services Web : Généralement, un diagramme d'interaction AUML peut être vu comme une séquence autorisée de messages entre agents et un ensemble de contraintes sur le contenu de ces messages. Il permet de modéliser les interactions entre agents utiles à l'implémentation du système. Cependant, il n'est pas suffisant pour capturer complètement la dynamique, notamment en ce qui concerne les aspects intra agent.

Pour remédier à ce manque, le langage BPEL4WS est utilisé pour offrir une spécification textuelle et plus détaillé des PI. En effet, d'autres informations supplémentaires peuvent êtres ajoutés particulièrement dans le cas où un ou plusieurs partenaire de l'interaction est un service Web. Pour invoquer un service Web par exemple (voir figure 2.3), l'opération WSDL invoquée est indiquée par l'attribut XML *operation* et associée aux informations *partnerLink* et *portType*. Le contenu envoyé peut provenir de la variable indiquée par l'attribut *outputVariable* et le contenu reçu peut être affecté à la variable indiquée par l'attribut *inputVariable*.

L'opérateur permet l'ajout de garde (*catch* et *catchAll*) en cas d'exception, et également de mécanismes de compensation en cas d'annulation d'une transaction (*compensationHandler*).

2.5 Formalisation des protocoles d'interaction avec les réseaux de Petri colorés

Cette partie consiste à utiliser un mécanisme formel pour exprimer la sémantique opérationnelle de notre modèle. A cet effet, nous définissons

les règles de transformation pour exprimer la sémantique des différentes constructions des langages de spécification des protocoles d'interaction. Le modèle formel retenu pour la représentation du comportement observable des processus métiers est les réseaux de Petri colorés (RdPC) [14]. Ce modèle présente certaines propriétés qui offrent les moyens de formaliser la notion d'une interaction correcte entre les processus métiers.

Le choix de ce formalisme est principalement motivé par l'existence d'une sémantique formelle et la richesse de techniques d'analyse, ce qui nous permet de définir des propriétés mathématiquement vérifiables. A ce stade, nous avons proposé un algorithme de synthèse qui, étant donnée la spécification d'un PI, permet de générer le RdPC correspondant en s'appuyant sur des règles de passage bien définies.

Lorsqu'un PI est opéré, nous nous plaçons dans l'hypothèse que les partenaires auront un comportement cohérent. Mais lorsque la structure d'interconnexion est modifiée, même si nous tenons compte de l'état interne des composants, il n'est pas évident de certifier que le système résultant se comportera correctement. Si les comportements ne sont pas compatibles entre eux, leur interaction peut introduire des inter-blocages.

Une stratégie souvent utilisée consiste à analyser le PI, avant de l'exploiter par le SMA. Elle implique l'utilisation de techniques de vérification comportementale de spécifications. Dans notre contexte, la cohérence de système est assurée par la définition d'un ensemble de règles pour les transformations garantissant le respect d'une certaine cohérence du système.

Un intérêt majeur de cette transformation est la richesse des outils de simulation existant (l'outil CPN-AMI par exemple), qui permettent de valider des propriétés comme la réception non spécifiée et l'absence d'interblocages.

Dans la suite, nous proposons un ensemble de règles de transformation des différentes notations des PI utilisées dans notre approche vers le formalisme de RdPC en utilisant une équivalence entre les éléments des deux modèles. Nous montrons d'abord comment on construit le RdPC en termes de structure et nous en donnons ensuite les domaines des couleurs, les variables, etc.

2.5.1 Dérivation des domaines de couleurs et de la structure de RdPC

Avant de créer le RdPC, il est nécessaire de définir la gestion des variables en utilisant des types de données et des domaines de couleurs. Un domaine de couleurs est associé aux places et aux transitions dans

le modèle réseau de Petri pour représenter l'information exigée afin de déterminer l'état du système.

En effet, nous considérons les domaines de couleurs et les variables suivantes :

Colour sets :
$Communicative\ Act = \textbf{with}\ inform|cfp|propose|\ldots;$
$Role = string\ \textbf{with}\ a\ldots z;$
$Content = string\ \textbf{with}\ a\ldots z;$
$Bool = \textbf{with}\ true|false;$
$MSG = record$
$\quad s, r : Role;$
$\quad CA : Communicative\ Act;$
$\quad C : Content;$
$Variables :$
$\quad msg1, msg2 : MSG;$
$\quad x : Bool;$

Le domaine de couleurs *MSG* décrit l'échange de messages dans une communication inter-processus. Dons ce cas, *MSG* est un enregistrement <s, r, ca, c>, où les éléments s et r déterminent l'expéditeur et le récepteur du message correspondant. Ces éléments ont le type *Rôle*, qui est utilisé pour identifier les processus métier ou/et les services Web participant dans l'interaction correspondante. Le type "Communicative Act" et *Contents* représentent respectivement les actes communicatifs FIPA-ACL et le contenu du message correspondant.

Nous montrons maintenant comment les différentes notations des PI décrites dans notre travail peuvent être représentées en utilisant le formalisme RdPC.

Pour cela, nous avons développé un ensemble de règles permettant ce passage. Les deux premières règles s'intéressent à la représentation des différents rôles de protocole ainsi que le mécanisme de communication (l'échange de messages durant l'interaction). Les autres règles traitent la représentation des différents types de messages (Parallélisme, choix exclusif ou inclusif) ainsi que celle des itérations.

Règle #1. Nous utilisons une approche par composition de réseaux, dans laquelle chaque processus métier ou/et services Web est représenté par les séquencements de places et de transitions appartenant à ce rôle. Comme le montre la Fig. 2.4, le réseau est donc constitué d'un sous-réseau pour chacun des rôles qui participe à l'interaction.

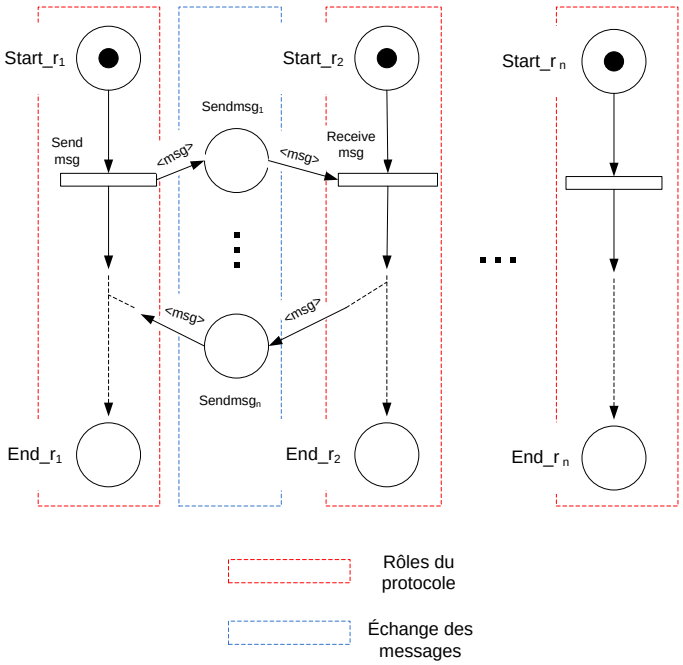

FIGURE 2.4 – *Structure générale du protocole d'interaction exprimée avec RdPC*

Chaque rôle r comprend une place $Start_r$ et une place End_r (non colorées et 1-bornées). Dès que $Start_r$ contient un jeton, cela entraîne le démarrage d'une exécution du rôle r. Ensuite, ces réseaux sont connectés par des places qui correspondent à l'échange des messages.

Règle #2. L'échange de messages entre deux rôles est représenté par une place de synchronisation et des arcs (Fig. 2.5). Le premier arc relie la transition d'envoi du message à la place de synchronisation et le deuxième relie cette place à la transition de réception.

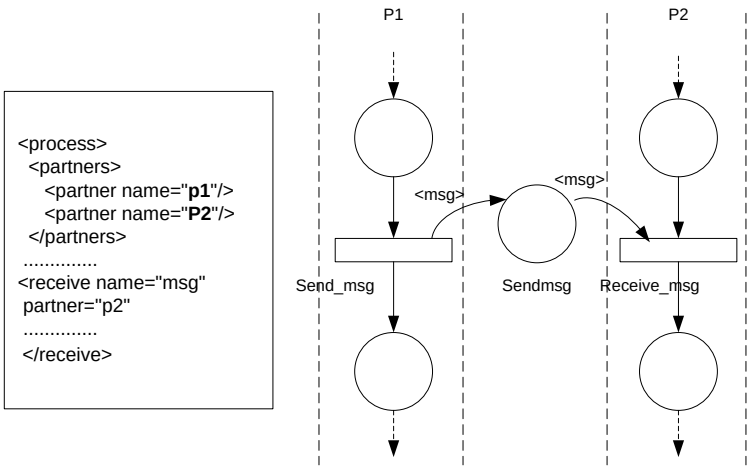

FIGURE 2.5 – *Échange des messages primitifs entre deux rôles*

Règle #3. Échange des messages primitifs : Comme le montre la défini-

tion d'un PI (section 3 de ce chapitre), un message primitif correspond au message simple. Les messages asynchrones (la section <*receive*> et <*reply*> de BPEL4WS) sont représentés par une place de synchronisation (voir Fig. 2.5) et les messages synchrones (la section <*invoke*>) par deux places de synchronisation. Cela permet d'exprimer le fait que ce type de communication ne s'achève que lorsque le message a été traité par le destinataire.

Règle #4. Échange des messages complexes : Un message complexe est composé à partir des messages simples (primitifs) par le moyen d'opérateurs. Dans le cas des PI, il existe trois types de messages complexes : messages de type : ou-exclusif, ou-inclusif et concurrents. Dans ce qui suit, nous allons présenter la traduction de ces trois types de messages.

Règle #4.1 Il s'agit du cas des messages de type ou-exclusif , où un et un seul message peut être envoyé à la fois. Le choix peut être fait entre différents messages. Ce type d'interaction est représenté par la section <*if*>/<*pick*> de BPEL4WS.

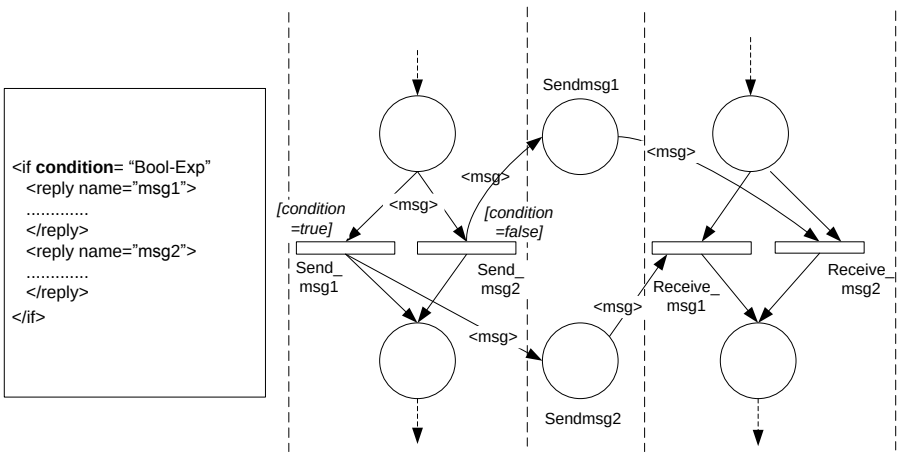

FIGURE 2.6 – *Échange de message complexe XOR*

La figure 2.6 montre une transformation possible de ce type d'interaction de sorte qu'exactement un acte de communication soit envoyé. Dans ce cas, pour chaque type de message on associe une transition avec une garde. Cette garde agit comme un filtre en ne permettant le franchissement de la transition qu'aux messages du type qui lui est définit.

Règle #4.2 Il s'agit des messages de type ou-inclusif pour lesquels zero, un ou plusieurs messages peuvent êtres envoyés en même temps. La figure 2.7 montre une transformation possible de ce type d'interaction.

Plus précisément, soit <msg_1 *or* msg_2 *or* ... *or* msg_n> un message complexe de type ou-inclusif. Cette règle permet alors de créer n transitions (nommée respectivement $Send\_msg_1$, $Send\_msg_2$, ..., $Send\_msg_n$)

avec n gardes ($garde_1$, $garde_2$, ..., $garde_n$). Pour exprimer aussi le cas où aucun message n'a été envoyé, nous ajoutons une transition supplémentaire (nommée *NoMsgSend*). Dans ce cas, la garde de cette transition est exprimée comme suite :

$$garde_{NoMsgSend} = \bigwedge_{i=1}^{n} \neg garde_i$$

Cela veut dire que cette transition est franchissable si les gardes des transitions $Sendmsg_i$ ne sont pas vérifiées (qui exprime qu'aucun message n'est envoyé).

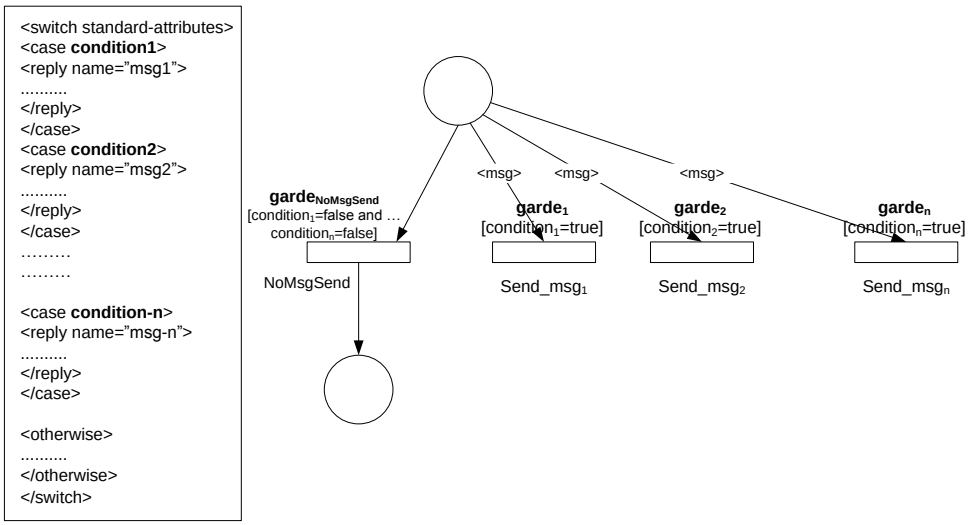

FIGURE 2.7 – *Échange de message complexe OR*

Règle #4.3 Il s'agit du cas où plusieurs messages sont envoyés en même temps. Dans ce cas et contrairement aux messages de types *OU* (inclusif ou exclusif), on a besoin d'une seule transition sans garde et n places de synchronisation (où n représente le nombre de messages à envoyer). Comme le montre la figure 2.8, cela veut dire que touts les messages sont envoyés en même temps.

Règle #5. La boucle d'une partie de la spécification BPEL4WS est représentée par la section <*while*> et une expression de condition. En RdPC la boucle est spécifiée par deux transitions (transition de début et de fin de la boucle) qui sont reliées pas une place et des arcs. La condition de la boucle est représentée par expression de garde (au niveau de la transition de début) qui est évaluée (voir Fig. 2.9).

Comme nous avons présenté précédemment, nous utilisons seulement les composants BPEL4WS qui permettent de modéliser l'interaction observée.

Ces composants sont regroupés en trois catégories :

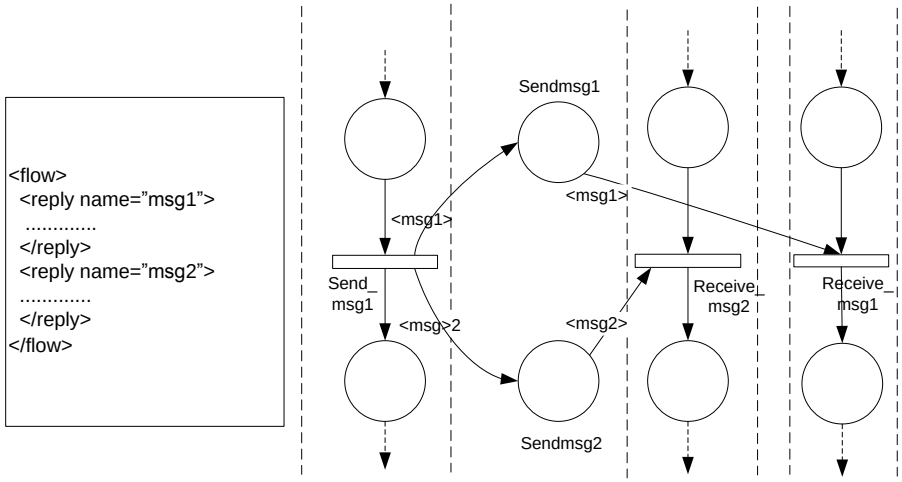

FIGURE 2.8 – *Échange de message complexe AND*

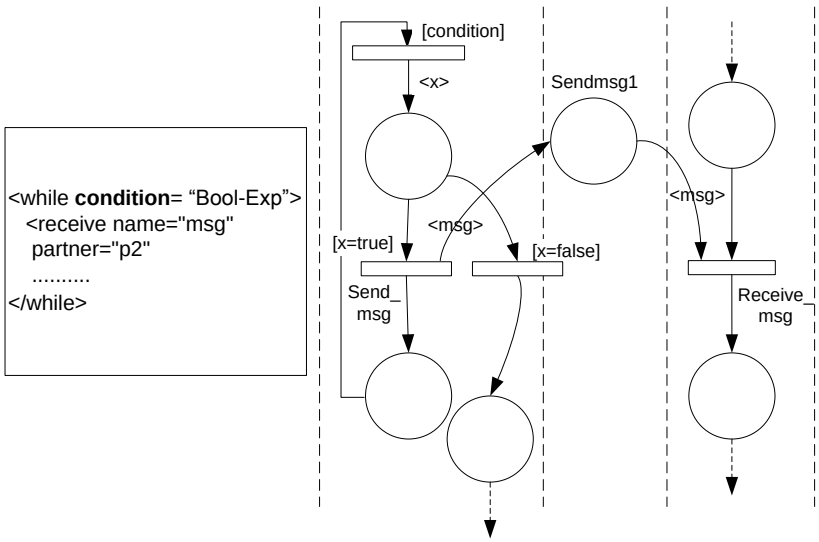

FIGURE 2.9 – *Iteration*

- Les composants de base tels que *partner*, la représentation RdPC de ces composants est assurée par la règle 1 ;
- Les composants basés sur les messages tels que *receive*, *reply* et *invoke*. Nous avons proposé les règles 2 et 3 ;
- Les composants de contrôle structurés et évolués tels que if, *pick*, *switch*, $flow$, *while*. Nous avons définir les règles 4 et 5.

En conséquence, les règles développées dans cette section permettent de couvrir touts les composants BPEL4WS que nous avons utilisés pour specifier les protocoles d'interaction.

2.5.2 IP2CPN : Un outil de génération de RdPC

Afin d'automatiser le passage des spécifications des PI vers les RdPC, nous avons développé l'outil IP2CPN. L'objet de cette section est de présenter le fonctionnement de cet outil afin de générer un RdPC à partir d'une spécification de PI. Le RdPC est construit en plusieurs itérations. En effet, l'algorithme IP2CPN permet de générer le reseau de Petri en explorant le protocole d'interaction. Les lignes 1 et 2 de l'algorithme (voir Fig. 2.10) initialisent les différentes variables utilisées ainsi que le RdPC résultat.

Comme le montre la règle 1, l'algorithme commence par la création des places de rôles, dénotées par la variable *RP*. Ces places correspondent aux rôles du protocole (la ligne 3, 4 et 5). Chacune de ces places est étiquetée par le nom de rôle correspondant.

Dans la boucle principale de l'algorithme (la ligne 7), la variable *curr* est initialisée au premier message de PI. Les lignes 8-16 permet de créer les composants de RdPC de l'itération actuelle. D'abord, à la ligne 8, les places de message, associées à la variable *curr* sont créées en utilisant la fonction *CreateMessagePlace*(). Ces places correspondent aux actes de communication.

Ensuite, nous créons les places intermédiaires (ligne 9) qui correspondent aux changements d'état d'interaction suite à l'envoi de messages associé à la variable *curr*. Ensuite et selon le type de message à traiter, les lignes entre 10 et 15 permet de connecter ces places par des transitions et des arcs en utilisant les fonctions *CreateTransitions*() et *CreateArcs*(). Par exemple, si la variable *curr* contient un message complexe de type or-exclusif alors la règle 4.2 est utilisée afin de connecter ces places par des transitions et des arcs.

Finalement, nous ajoutons les couleurs des jetons à la structure de RdPC, en utilisant la fonction *FixColor*() (la ligne 16).

Pour compléter la boucle, la structure de RdPC est mise à jour selon l'itération actuelle (lignes 17-19). La boucle réitère tant que *M* contient des

Algorithm $CreateIP - net$ (input : IP=$< ID, R, M, f_M >$, output : CPN)

1: RP $\leftarrow \phi$ // Roles places
   MP $\leftarrow \phi$ // Messages places
   IM $\leftarrow \phi$ // Intermediate places
   TR $\leftarrow \phi$ // list of transitions
   AR $\leftarrow \phi$ // list of arcs
2: CPN $\leftarrow$ new CPN
3: For every $r \in R$ do
4: RP $\leftarrow$ $createRolePlace()$ // there would be one token in every RP place
5: CPN.places $\leftarrow$ RP
6: While M $\neq \phi$ do
7: curr $\leftarrow$ M.dequeue()
8: MP $\leftarrow$ $CreateMessagePlace(curr)$
9: IM $\leftarrow$ $CreateIntermediatePlace(curr, MP)$
10: TR $\leftarrow$ $CreateTransitions(curr, MP, IM)$
11: If curr.CA $\notin$ {Failure, Cancel, not-understood}
12: AR $\leftarrow$ $CreateArcs(curr, MP, IM, TR)$
13: Else // MP is a terminating place
14: AR $\leftarrow$ $CreateArcs(curr, IM, TR)$
15: End If
16: $FixColor(MP, TR, AR, curr.CA)$
17: CPN.places $\leftarrow$ CPN.places $\bigcup$ MP $\bigcup$ IM
18: CPN.transitions $\leftarrow$ CPN.transitions $\bigcup$ TR
19: CPN.arcs $\leftarrow$ CPN.arcs $\bigcup$ AR
20: End while
21: Return CPN

FIGURE 2.10 – *Présentation de l'algorithme IP2CPN*

messages qui n'ont pas encore traités. Finalement, le RdPC résultant est retourné (la ligne 21).

Complexité de l'algorithme

La complexité d'un algorithme est le nombre d'opérations élémentaires (affectations, comparaisons, opérations arithmétiques) effectuées par un algorithme. Ce nombre s'exprime en fonction de la taille n des données. On s'intéresse au coût exact quand c'est possible, mais également au coût moyen (que se passe-t-il si on moyenne sur toutes les exécutions du programme sur des données de taille n), au cas le plus favorable, ou bien au cas le pire. On dit que la complexité de l'algorithme est $O(f(n))$ où f est d'habitude une combinaison de polynômes, logarithmes ou exponentielles. Ceci reprend la notation mathématique classique, et signifie que le nombre d'opérations effectuées est borné par $cf(n)$, où c est une constante, lorsque n tend vers l'infini.

Dans notre cas, l'algorithme proposé est de complexité linéaire $O(n)$. Plus précisément, pour générer le RdPC , on effectue $6 + n$ affectations où n est le nombre de rôles du protocole. Dans la boucle principale de l'algorithme, on effectue $9m$ affectations où m représente le nombre de

messages dans le protocole. La complexité de $6 + n + 9m$ est $O(n)$, donc la complexité de l'algorithme IP2CPN est $O(n)$.

2.6 Vérification et Validation

Un protocole invalide ne pourra pas garantir une interaction fiable. C'est pourquoi il est important d'analyser et de valider les protocoles d'interaction pour prévoir et prévenir s'ils sont susceptibles de fautes comme des interblocages ou des ambiguïtés.

La méthode de validation qui est la plus utilisée aujourd'hui, est l'analyse d'accessibilité. Elle consiste à construire le graphe global de l'interaction où chaque état correspond à un état de la communication. Les états et la structure de ce graphe sont ensuite examinés à la lumière d'un certain nombre de propriétés (interblocage, réception non spécifiée, etc.), pour valider le protocole d'interaction mis en oeuvre.

2.6.1 Analyse d'accessibilité

Cette technique consiste à générer le graphe des états accessibles. Dans notre contexte de travail, chaque état du graphe représente un état de l'interaction tenant compte des valeurs des différentes variables utilisées par le protocole. La génération du graphe des états accessibles correspond à un parcours en profondeur de la spécification du protocole.

Comme tout autre langage formel, l'utilisation pratique du modèle de RdPC est possible grâce à l'existence d'outils adéquats. Dans ce travail, nous avons utilisé l'outil CPN-AMI [23] développé au LIP6. Cet outil est dédié à la théorie des réseaux de Petri colorés et supporte la spécification, la validation, la vérification formelle. Les services qu'il offre permettent de vérifier des formules de logique temporelle (model checking), de calculer des propriétés structurelles (invariants, verrous, trappes), de simuler, de déboguer et d'engendrer du code [4]. L'accès aux outils se fait à travers une interface utilisateur (Coloane [22]) qui est un éditeur de graphes génériques.

2.6.2 Propriétés vérifiées

Dans notre travail, nous nous intéressons à deux types de propriétés : des propriétés générales que l'on retrouve dans tous les protocoles comme les problèmes d'interblocage, de réception non spécifiée et des propriétés fonctionnelles qui sont dépendantes de la fonction du protocole et qui sont décrites dans le cahier des charges lors de la phase d'analyse.

Propriétés générales

Les propriétés générales représentent les conditions de base de la validité des spécifications des protocoles. Ces propriétés peuvent être résumées comme suit :

1. **Absence d'interblocage :** Dans notre modélisation, le RdP vérifie cette propriété si et seulement si tous les rôles impliqués dans une interaction atteignent leurs états de fin respectifs. Plus précisément, soit End_r la place finale pour chaque rôle r impliqué dans l'interaction et soit *endsta* l'ensemble des états terminaux du graphe des états accessibles.

$$\forall r \in R, End_r \in endsta \Leftrightarrow the\ net\ is\ deadlock\ free \tag{2.1}$$

Cette formule montre que chaque rôle r atteint son état final étiqueté par End_r. Cependant, un rôle r qui aboutit à un état $state_r$ où, $state_r$ $\neq$End$_r$ représente un rôle actif : la terminaison dans un état où les rôles sont actifs indique une violation. Cette formule est représentée par la requête CTL suivante :

$$\forall r \in R, End_r \in endsta, AF(card(End_r) > 0) \tag{2.2}$$

Notons ici que $AF(\Phi)$ signifie que pour toute exécution, il existe un état où (Φ) est vérifiée (se référer à la section 1.4.2 de chapitre 1). La fonction $card(x)$ donne le nombre de jetons dans la place x, $card(x) > 0$ signifie que la place x est marquée.

Cependant, et en raison de la sémantique indéterminisme de messages complexes de type OR/XOR. Cette formule n'est pas satisfaite pour chaque $r \in R$. Pour mieux expliquer cette situation, soit l'exemple du protocole d'interaction exprimé avec AUML dans la Fig. 2.11-a. Dans cet exemple, le rôle r_1 envoi un message complexe de type $or - inclusif$ aux rôles r_2 et r_3. Enfin, le rôle r_3 envoi un message simple au rôle r_2. Après l'application des règles 4.2 et 3, on obtient le RdPC de la Fig. 2.11-b.

Nous rappelons que dans le cas de message de type or-inclusif, zero, un ou plusieurs messages peuvent êtres envoyés en même temps. Nous prenons par exemple le premier cas, (c-à-d aucun message n'est envoyé). Cette situation implique le franchissement de la transition *NoMsgSend*. Nous remarquons qu'aucune transition n'est franchissable à partir de ce cas (les sous-réseaux r_1 et r_2 ne progressent plus) et par conséquent, le message *request* ne sera jamais envoyé.

Ce problème est dû à l'indéterminisme de messages complexes de type OR/XOR.

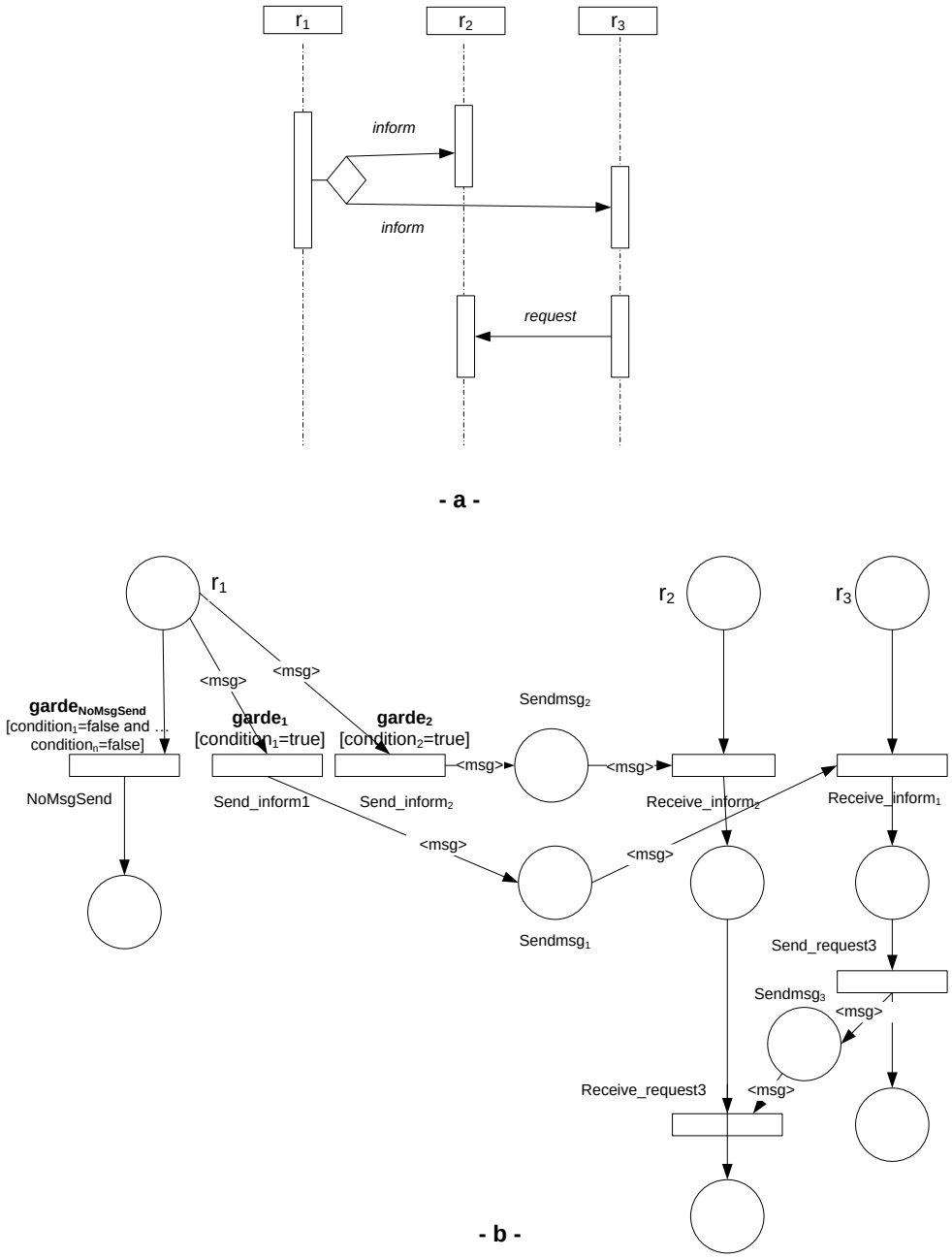

FIGURE 2.11 – *Problème de l'indéterminisme de messages complexes de type OR/XOR*

En effet et pour remédier à ce problème (et par conséquent assurer que la formule 2.1 est satisfaite), chaque sous-réseau est équipé d'une itération. Donc nous ajoutons une transition et deux arcs de la place de fin à cette transition et de cette transition à la place de début. Nous notons qu'avant l'application de la formule 1, il est important de vérifier que le réseau est borné.

2. **La vivacité :** tous les état sont atteignables à partir de l'état initial

et la consommation de tous les messages du protocole est assurée. Cette propriété est capturée par la formule CTL suivante :

$$\forall i \in [1,n], AG(IfThen(card(Sendmsg_i) > 0, \\ AF(card(Sendmsg_i) == 0)))$$

Nous rappelons ici que *Sendmsg* correspond aux places de synchronisation qui représentent l'échange de messages entres les différents rôles du protocole (voir la Fig. 2.4). Aussi, $AG(\Phi)$ signifie que pour toute exécution, Φ est toujours vérifié.

3. D'autres propriétés peuvent être vérifiées aussi longtemps qu'elles sont concernées par des états des rôles et de leur comportement. Par exemple la requête CTL suivante vérifie que dans un message complexe de type XOR, un et seulement un seul message, peut être envoyé :

$$\forall i,j \in [1,m] \, and \, i \neq j, \, AG(IfThen(card(Sendmsg_i) > 0, \\ card(Sendmsg_j) == 0))$$

Propriétés spécifiques aux protocoles d'interaction

Les techniques de vérification que nous avons présentés dans la section précédente ne se focalisent que sur la cohérence du système. Elles ne garantissent pas que le protocole implémenté répond réellement aux attentes des concepteurs.

Les propriétés spécifiques aux protocoles dépendent de domaine d'application et sont décrites dans le cahier des charges lors de phase d'analyse. Ces propriétés peuvent être exprimées avec la sémantique de logique temporelle (la logique CTL dans notre cas).

Un PI, pour être utilisable, doit vérifier un certain nombre de propriétés bien spécifiques. Dans notre approche, ces propriétés interviennent aussi bien dans les dialogues inter-processus.

Par exemple la requête CTL suivante permet d'exprimer le fait qu'une communication synchrone ne s'achève que lorsque le message a été traité par le destinataire.

$$AG(IfThen(card(Sendmsg_n : (field[1] == r_2)) > 0, \\ AF(card(Sendmsg_m : (field[1] == r_1)) > 0)))$$

2.7 Conclusion

Nous avons proposé dans ce chapitre une formalisation des protocoles d'interaction appliqués au domaine d'intégration d'applications par coordination des processus.

En effet, nous avons présenté d'abord les fondements de la notation AUML pour la modélisation des interactions des processus métiers. Cette approche permet de définir un modèle abstrait d'interaction entre collaborateurs participant à une activité de l'entreprise, voire entre une organisation et ses partenaires.

Nous avons défini par la suite des règles permettant de passer à partir d'une description semi-formelles des protocoles d'interaction vers des spécifications formelles RdPC tout en mettant l'accent sur le contrôle et la dynamicité de l'interaction. Ce chapitre se termine par la présentation d'un algorithme pour la traduction des protocoles en RdPC. Cette traduction permet d'envisager l'utilisation d'outils tiers pour la vérification.

Le chapitre suivant présente notre architecture d'intégration afin de permettre aux agents de pouvoir utiliser ces protocoles.

Une architecture basée agent pour l'intégration d'applications

3

Sommaire

Les systèmes multi-agents offrent de nombreux avantages pour la conception et le contrôle des systèmes complexes. Cependant leur développement reste difficile. Les difficultés de développement inhérentes aux systèmes multi-agents, sont la conséquence de la diversité et de la complexité des concepts d'agent et des mécanismes de coordination, d'interaction, d'organisation, etc. Cette complexité fait que l'utilisation de la plupart des outils et des plates-formes existants est difficile aux non-spécialistes en multi-agents [68].

Dans ce chapitre nous nous intéressons aux systèmes multi-agents dont l'interaction se base sur les PI définis dans le chapitre précèdent. Le développement de ces systèmes nécessite une bonne maîtrise des protocoles d'interaction et de tous les détails de leur utilisation. Le développeur

doit connaître le langage de communication à utiliser, tels que FIPA-ACL, l'enchaînement des messages et la spécification de leur contenu. De ce fait, l'implémentation des protocoles d'interaction est complexe, de même pour l'implémentation des agents qui les utilisent.

Cette problématique nous a conduit à concevoir et implementer une architecture basée agent pour l'intégration des processus métiers (figure 3.1). Cette dernière inclut une bibliothèque de protocoles d'interaction implémentés d'une manière générique, et propose un framework d'agents interactifs.

Ce chapitre est dédié à la présentation de notre architecture d'intégration. Nous commençons la description de notre architecture par la motivation de l'approche agent. Ensuite, nous présentons les structures internes des différents agents impliqués, ainsi que leurs rôles dans le processus d'intégration.

3.1 Motivations de l'approche agent

La technologie multi-agents a attiré davantage d'attention grâce à l'évolution d'Internet qui intègre une infrastructure représentant une organisation d'espace dans laquelle des agents autonomes opèrent et interagissent les uns avec les autres [100][85]. En effet, L'orientation du développement de notre architecture d'intégration vers l'approche agent est motivée par les points suivants :

- La nature du domaine d'intégration d'applications par coordination des processus (ensemble des processus métiers autonomes, géographiquement dispersés et voulant coopérer pour réaliser un but commun) s'adapte bien à l'approche " systèmes multi-agent répartis ", et par conséquent, le mapping entre les deux technologies (BPI et SMA) se fait de manière naturelle.
- L'exécution efficace et la supervision des processus métiers répartis exigent des réactions rapides de la part des entreprises membres. Les réseaux sont les médias privilégiés pour la communication, de ce fait, le besoin que chaque entreprise possède un " représentant " dans le réseau. Cela peut être matérialisé à l'aide de la notion d'agent.
- L'introduction de ce paradigme dans le domaine d'intégration d'applications permet de bénéficier des solutions apportées par les travaux de recherche dans le domaine de l'IA et de l'agent. Par exemple, pendant la coordination des processus métiers dans laquelle il est nécessaire de sélectionner des partenaires et de distribuer des tâches, montre le besoin de certaines caractéristiques du système telles que le besoin de négociation. Ces points ont été discutés profondément dans des travaux de recherche portant sur les SMAs.
- Le mécanisme d'adaptabilité des SMAs semble être adéquate pour l'intégration dynamique dans laquelle, différents niveaux de coordination avec des ensembles différents de partenaires pourraient être établis dans les différentes phases.
- L'infrastructure SMA est un ensemble de services, de conventions et de connaissances supportant les interactions sociales complexes des agents. La coordination et la résolution des problèmes distribués sont des problèmes critiques pour l'intégration d'applications. Néanmoins, ils peuvent trouver des solutions acceptables dans une approche multi-agent.
- Une approche multi-agent fournit une bonne solution pour la mise en oeuvre du scénario d'intégration, en tenant compte des questions sensibles qui doivent trouver des solutions satisfaisantes, à savoir :

la coordination des processus métiers, le maintien de l'autonomie et la confidentialité des données.

3.2 Quelques modèles et architectures de coordination des processus

L'intégration d'applications par coordination des processus est un problème fondamental dans le développement des systèmes d'entreprises. Elle est devenue un champ de recherche à part entière. La répartition, l'autonomie et l'hétérogénéité, qui font partie de la nature des services d'information actuels, posent des défis techniques sévères pour les organisations modernes quant à la génération d'information cohérente et compacte [51][141].

Plusieurs architectures et frameworks traitant la coordination des processus dans les environnements ouverts ont été développés par la communauté des chercheurs. Nous présentons dans cette section ceux qui se rapprochent l'architecture de coordination que nous proposons dans ce travail et qui est basé sur la notion d'agent.

Licci et al. [6] ont proposé une approche dont le principe semble similaire à celui utilisé dans notre architecture. L'intégration des processus et la gestion des workflows sont modélisés comme des problèmes de coordination. Le modèle et la technologie TuCSoN ont été construits pour la coordination des agents. La coordination objective, i.e., coordination à l'extérieur des agents, a été utilisée. Elle fournit un support effectif pour les trois propriétés suivantes : dynamicité, flexibilité et hétérogénéité.

Une autre approche intéressante pour supporter le mécanisme de la coordination dans les SMA est la formation de coalitions. Les coalitions sont des organisations qui fournissent une dynamique et une flexibilité pour pouvoir supporter les situations de coopération et de compétitivité dans les environnements ouverts [53].

Plusieurs travaux exploitant les méthodologies orientées agent et SMA pour la coordination des processus proposent d'utiliser des mécanismes de formation de coalitions. L'approche de [130] propose par exemple une approche basée sur la composition des services à travers la coalition d'agents médiateurs qui ont pour rôle de réaliser les tâches requises par les processus métiers. Les besoins à satisfaire sont exprimés en termes de contraintes et de préférences. La décomposition de ces besoins est ensuite faite au niveau des agents médiateurs qui consultent une ontologie de tâches [8] pour déduire qu'une tâche à réaliser est complexe et implique la réalisation de plusieurs sous-tâches.

Les approches proposées de coordination des processus par formation de coalitions utilisent des métriques d'évaluation quantitatives telles que

le nombre de coalitions formées et abandonnées et négligent l'évaluation qualitative du résultat fourni par la composition de services en fonction de besoins donnés à satisfaire [41].

Synthèse

A l'image de cette étude, il est clair que plusieurs travaux de recherche ont exploité l'approche agent pour la coordination des processus métiers. Le développement de ces systèmes nécessite une bonne maîtrise des protocoles d'interaction et de tous les détails de leur utilisation. Le développeur doit connaître le langage de communication à utiliser, tels que FIPA-ACL, l'enchaînement des messages et la spécification de leur contenu. De ce fait, l'implémentation des protocoles d'interaction est complexe, de même pour l'implémentation des agents qui les utilisent.

Cette problématique nous a conduit à concevoir et implementer une architecture basée agent pour l'intégration des processus métiers (figure 3.1). Cette dernière possède une bibliothèque de protocoles d'interaction implémentés d'une manière générique, et propose un framework d'agents interactifs.

Dans le contexte de notre travail, le mécanisme de coordination est mise en oeuvre à l'aide d'un agent intermédiaire (i.e., encapsulation des protocoles d'interaction directement à l'intérieur de cet agent). Il fournit également un support effectif pour les trois propriétés précédentes. Le fait d'utiliser une coordination objective permet d'appliquer d'une manière effective la collaboration des agents distribués dans la gestion des processus métiers, où l'activité de contrôle de haut niveau est assurée par l'agent intermédiaire, que nous l'appelons agent *Intégrateur*.

Dans la suite de ce chapitre, nous présentons notre architecture d'intégration en terme des structures internes des différents agents, ainsi que leurs rôles dans le scénario d'intégration.

3.3 Description architecturale

Généralement, une architecture exprime la structure fondamentale du système à analyser et à concevoir. L'architecture définit un ensemble de composants fonctionnels, des sous-systèmes ou de modules décrits en termes de leurs comportements et interfaces [101] [129]. Elle définit aussi comment ces composants interagissent et comment ils doivent être interconnectés pour accomplir correctement les buts du système. Ainsi, une description architecturale est principalement requise pour la spécification de la structure du système.

En principe, le terme composant inclut les éléments fonctionnels qui peuvent être des objets répartis ou bien des agents autonomes tel est le cas

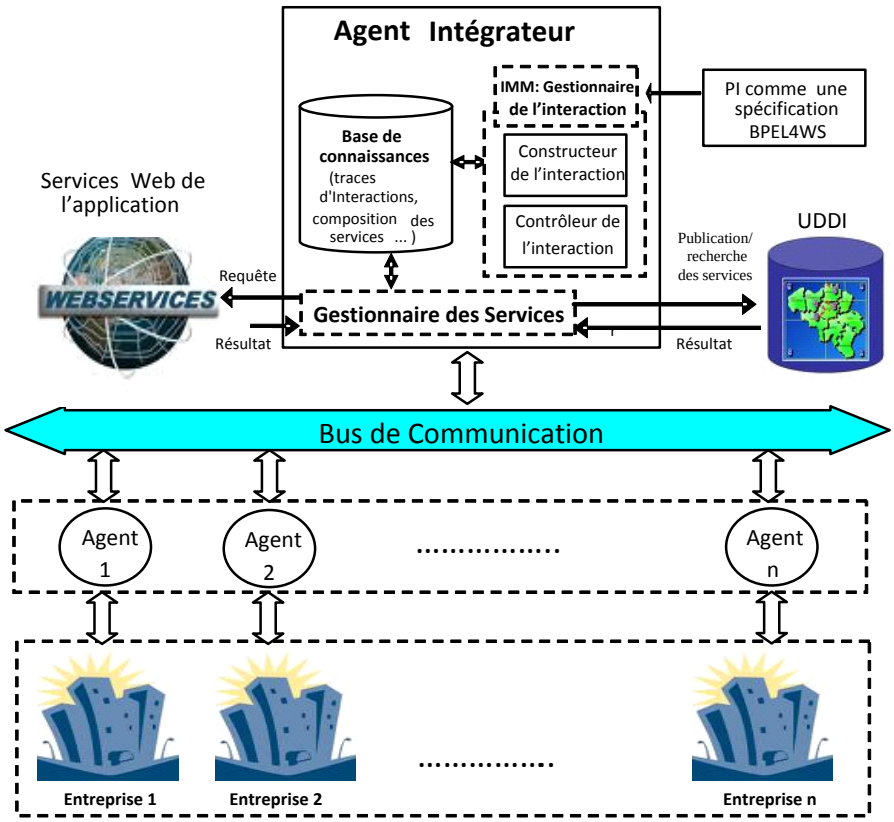

FIGURE 3.1 – *Architecture du système proposé*

dans notre travail. En prenant en considération ces définitions, nous allons présenter dans les sections suivantes de ce chapitre les spécifications des différents composants, ainsi que les concepts liés à leur fonctionnement.

3.3.1 Aperçu de l'architecture proposée

L'architecture d'intégration que nous proposons dans ce travail est basée sur un framework constitué d'agents interactifs et une bibliothèque de composants réutilisables.

- Le framework d'agents fournit les composants de base et une structure minimale d'agents interactifs que le développeur pourra réutiliser pour développer ses agents.
- La bibliothèque contient un ensemble de protocoles d'interactions dont les rôles sont implémentés sous forme de composants. Cette bibliothèque est gérée par l'agent Intégrateur. L'ensemble de ces protocoles correspond à l'ensemble des stratégies dont l'agent Intégrateur dispose pour atteindre ses objectifs.

Comme nous l'avons mentionné précédemment, les entreprises sont représentées par des agents autonomes, géographiquement répartis, et qui

peuvent coopérer pour réaliser un objectif métier commun. L'idée principale consiste à ajouter un agent intermédiaire nommé «Agent Intégrateur» (figure 3.1) entre le SMA et sa partie d'interaction exprimée avec le langage BPEL4WS. Cet agent permet de gérer l'interaction en interprétant la description des protocoles.

Le déroulement d'une interaction met en jeu des interventions effectuées par les participants à cette interaction, ainsi qu'un contrôleur vérifiant que ces interventions respectent les règles d'interaction. La réalisation des interventions est exclusivement à la charge des entités participantes. Cependant, il n'y aucune nécessité à ce que le code de synchronisation qui traduit le contrôle soit aussi réparti entre ces entités. Pour cela, un type d'agent est défini pour le contrôle et le suivi d'interaction.

L'idée de créer un agent intermédiaire pour le contrôle et le suivi d'intaraction offre beaucoup d'avantages, notamment, la facilité de la validation, la réutilisation, la maintenance des protocoles d'interaction et le non encombrement des agents avec des règles d'interaction.

Dans le section suivante, nous présentons la structure interne des différents types d'agents de notre architecture.

3.3.2 Structure de l'agent "Intégrateur"

Pour implémenter notre architecture, nous reprenons la recommandation du W3C définissant l'architecture d'un service web comme suit :

> "Un service web est vu comme une notion abstraite qui doit être implémentée par un agent concret. L'agent est une entité concrète (logicielle) qui envoie et reçoit des messages, alors que le service est l'ensemble de fonctionnalités offertes" [31].

Dans cette définition, un service est vu comme partie d'un agent. De même, notre approche d'intégration utilise un modèle de coordination se basant sur des agents offrant des services. Comme le montre la figure 3.1, l'agent Intégrateur est composé de quatres principaux modules :

- Le module de connaissances inclut une base de données et un ensemble d'actions.

  Le rôle de ce module est de mémoriser les informations qui sont nécessaires pour le bon déroulement de l'interaction. Parmi ces informations se trouvent les traces d'interaction utilisées durant les phases de coordination et les états courants des tâches en cours d'exécution. Ces informations sont utiles pour une éventuelle réutilisation du protocole.
- Le module de gestion de l'interaction IMM (pour Interaction Manager Module). Ce module permet le traitement des messages en

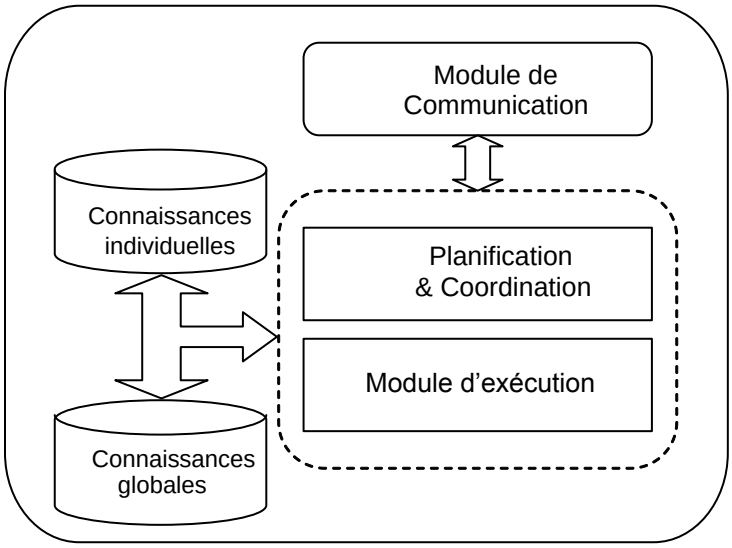

FIGURE 3.2 – *Structure d'un agent d'entreprise*

fonction de la specification des protocoles d'interaction (la specification BPEL4WS) et la partie connaissances.

- Le module de gestion des services offre les services nécessaires pour la localisation et l'invocation de services.
- La couche de communication permet le transport des messages. Dans notre travail, nous utilisons le langage FIPA-ACL [44] comme langage de communication entre les différents agents. Celui-ci a pour rôle de structurer les messages construits par l'agent (voir chapitre 1). Par ailleurs, les agents peuvent communiquer sur le web via le protocole de transport standard SOAP.

3.3.3 Structure de l'agent d'entreprise

L'agent d'entreprise (l'agent participant) est représenté par une structure comportant un module de communication, un module de planification et de coordination et un module d'exécution. Cet agent possède deux bases de connaissances, qui sont : les connaissances individuelles et les connaissances globales (figure 3.2)

- Le module de communication : contient tous les processus de prise en charge des messages, à savoir la réception, le filtrage et la traduction des messages entrants et la formulation et l'envoi des messages sortants.
- Le module de planification et de coordination : il est responsable de la gestion de la coopération et la formulation des offres afin de répondre aux sous-buts (les parties de la décomposition du but global) annoncés par l'agent Intégrateur,

```
<?XML version='1.0' encoding='UTF-8'?>
<!ELEMENT Agent_State (Goal+,Competence)>
<!ATTLIST Agent_State EAg_id CDATA #REQUIRED>
<!ELEMENT Goal(Goal_id, Comp_id, IP_id)*>
<!ELEMENT Goal_id (#PCDATA)>
<!ELEMENT Comp_id (#PCDATA)>
<!ELEMENT IP_id (#PCDATA)>
<!ELEMENT competence (Comp_id, Des, Price_h)+>
<!ELEMENT Comp_id (#PCDATA)>
<!ELEMENT Des (#PCDATA)>
<!ELEMENT Price_h (#PCDATA)>
```

FIGURE 3.3 – *Représentation d'un état d'agent en XML*

- Le module d'exécution : ce module contient les informations sur les ressources internes de l'entreprise (application, usagers, sources de connaissances, etc.) qui permettent la réalisation des tâches locales assignées à cette entreprise. Ce module a le rôle de réaliser la correspondance entre le sous but (la tâche) assigné à l'agent et les ressources internes de l'entreprise aptes à l'achèvement de cette tâche.
- La base de "Connaissances globales" : contient les informations concernant l'état de l'interaction dont laquelle il participe ainsi que la représentation des autres agents intervenant dans cette interaction.
- La base "connaissances individuelles" : contient des informations sur l'agent lui-même, c'est-à-dire ses capacités et ses compétences, l'état et la charge de travail actuelle, ... etc.

La structure présentée dans la figure 3.3 est un exemple d'informations sur l'état d'un agent d'entreprise (contenues dans le module 'connaissances individuelles'). La structure est décrite en XML dans la partie DTD (Document Type Definition).

Dans cette structure l'élément *Agent-State* exprime l'état de l'agent d'entreprise. Elle comprend des informations sur les buts de l'agent en question et un ensemble des compétences que possède cet agent. Pour chaque tâche, les compétences requises sont associées.

3.3.4 Bibliothèque des protocoles d'interaction

Dans l'architecture du système que nous proposons, se trouve une bibliothèque des PI. Cette bibliothèque est gérée par l'agent Intégrateur qui transmet les rôles aux agents participants.

Afin de clarifier la notion de la bibliothèque des protocoles d'interaction nous introduisons quelques aspects de son implémentation. Les principaux attributs d'un protocole que nous considérons sont :

- l'identifiant de l'interaction,
- le nom du rôle et le nom du protocole,
- un ensemble non vide de messages propres au protocole,
- l'état final de l'interaction qui peut avoir la valeur "succès" ou "échec". Cette information est retournée à l'agent Intégrateur à la fin de l'interaction. Sur la base de cette information, l'agent peut entreprendre certaines actions, tels que l'enregistrement du résultat de ses interactions ou la décision de s'engager dans une nouvelle interaction.

Ces informations sont extraies à partir du document BPEL4WS (la partie PI dans la figure 3.1). Pour une éventuelle réutilisation de ce protocole d'interaction, ces informations sont stockées dans la base de connaissances de l'agent Intégrateur.

3.4 Rôles et comportements des agents

La spécification d'un PI implique qu'on doit préciser la sémantique des messages échangés. On y décrit aussi les réactions d'un agent à un message donné. Dans notre cas, deux principaux rôles interviennent dans les protocoles d'interaction :

1. L'agent Intégrateur prend en charge l'extraction des données à partir de la spécification du protocole d'interaction et de les diffuser aux agents participant à l'interaction.
2. Les agents participants ont pour rôle de se coordonner afin de satisfaire l'ensemble des requêtes de l'agent Intégrateur. Le protocole d'interaction que nous spécifions dans ce chapitre décrit les règles utilisées par les agents participants pour se coordonner. Il y est également précisé les conditions sous lesquelles les agents participants envoient leurs réponses à l'agent Intégrateur.

Le comportement des agents, illustré dans la figure 3.4, peut alors être synthétisé comme suit :

1. L'agent Intégrateur (l'initiateur du processus d'intégration) annonce le début du processus d'intégration et l'allocation des tâches, en fai-

sant sortir une annonce destinée aux participants inclus dans le protocole d'interaction. Pour chaque tâche à réaliser, l'agent Intégrateur annonce une spécification qui comporte sa description, ainsi qu'une liste des exigences et des contraintes relatives à cette tâche. Ces informations sont encapsulées dans un message, devenant un message initiateur (m_0 sur la figure 3.4), qu'il diffuse à tous les agents participants).

2. Une fois que les agents participants sont identifiés, la phase d'allocation des sous-buts aux agents participants commence. Chaque participant (agent) réalise la partie qui lui a été assignée. La composition des résultats individuels aboutit à la réalisation du but global.
3. L'agent Intégrateur assure la coordination au niveau inter-agents (inter-entreprises) en respectant les règles d'interaction définies dans la specification BPEL4WS. Dans le contexte de ce travail nous n'aborderons pas l'aspect de coordination locale, et qui peut être différent d'une entreprise à une autre.

Pour garantir la bonne fin de l'interaction, il faut que l'agent Intégrateur sache à combien de réponses il doit s'attendre de la part des agents participants. L'analyse des règles d'interaction définies dans la specification BPEL4WS ainsi que la sémantique de l'ACL lui permettent alors de savoir si d'autres messages sont susceptibles de lui parvenir.

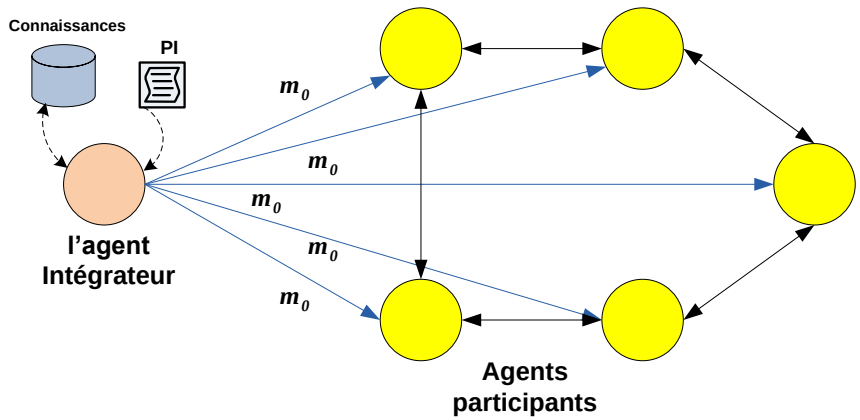

Figure 3.4 – *Comportement des agents*

3.5 Modes de communication

Les agents doivent communiquer aussi bien entre eux qu'avec les services de plate-forme ou de l'environnement. Dans notre architecture, il existe trois modes de communication :

1. Communication Agent-Agent : elle se produit via le langage de communication agent de FIPA-ACL, et elle est matérialisée par le système de gestion agent (AMS [1] de FIPA)
2. Communication Agent-service Web : elle est accomplie par des messages SOAP.
3. Communication Agent-Espace de données partagées : elle utilise les protocoles/interfaces appropriés fournit par l'espace de données. Ce dernier est utilisé pour stocker les spécification BPEL4WS qui maintient l'état du processus public.

3.5.1 Communication inter-agent

Le module de communication gère la réception et l'interprétation des messages provenant des autres agents. Cette gestion concerne le médium linguistique de la transmission des messages, c'est-à-dire l'expression et l'interprétation d'un message. Nous avons retenu, comme langage d'expression des messages le langage de communication ACL de la FIPA [44]. ACL présente sur KQML [2] [127] l'avantage d'une sémantique plus rigoureuse ainsi qu'un effort important de standardisation.

Les messages échangés

FIPA fournit des "performatifs" basés sur l'acte de discours et une syntaxe standard pour les messages. Ces messages sont basés sur la théorie des actes de discours, qui est le résultat de l'analyse linguistique de la communication humaine. La base de cette théorie est de produire une action à partir du langage. L'exemple dans la figure 3.5, présente la structure d'un message ACL :

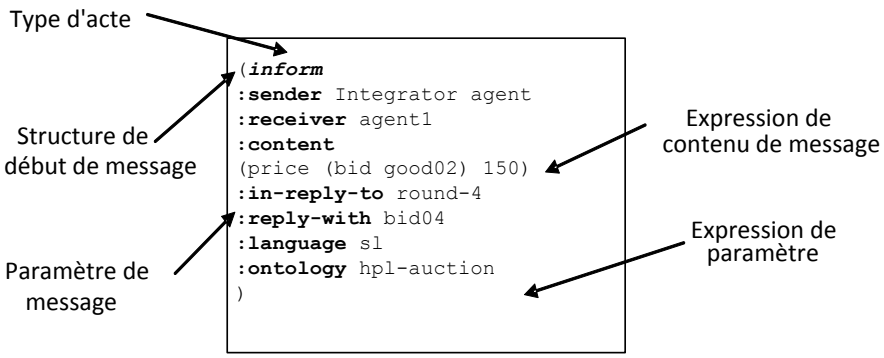

FIGURE 3.5 – *Structure d'un message ACL*

La signification sémantique des paramètres d'un message est la suivante [44] :

1. AMS : Agent Management System
2. Knowledge Query and Manipulation Language

Sender : Dénote l'identité de l'expéditeur du message ;

Receiver : Dénote l'identité du destinataire du message qui pourrait être le nom d'agent simple, ou un tuple de noms d'agent.

Content : Dénote le contenu du message ;

Reply-with : Présente une expression qui sera employée par l'agent répondant à ce message pour identifier le message original.

In-reply-to : Dénote une expression qui fait référence à une action précédente à laquelle ce message répond.

Language : Dénote le codage du contenu de l'action.

Ontology : Dénote l'ontologie employée pour donner la signification (le sens) des symboles dans l'expression.

Le contenu de messages

Dans le langage FIPA-ACL, aucun langage spécifique pour la description des contenus des messages n'est imposé. Plusieurs langages peuvent être utilisés tels que le langage KIF (Knowledge Interchange Format), le langage sémantique (SL) et Le langage XML (Extensible Markup Language).

Nous utilisons le langage XML pour la description, la spécification et l'interprétation du contenu des messages (Content) échangés entre les différents agents. Donc, les messages échangés entre les agents sont décrits à l'aide de FIPA-ACL et XML.

Le contenu de messages envoyés par l'agent Intégrateur aux différents agents d'entreprise est la description du sous but (la tâche) annoncé. En conséquence, les messages envoyés par les différents agents à l'agent Intégrateur contiennent la description (en XML) de leurs réponses. L'utilisation d'XML pour le contenu des communications entre agents permet de faciliter l'intégration avec les applications Web existantes.

3.5.2 Communication Agent - service Web

Ce type de communication est assuré par le module *Gestionnaire des services* de l'agent Intégrateur. Ce module offre les services nécessaires pour la localisation et l'invocation de services. Son fonctionnement est présenté comme suit (voir fig. 3.6) :

1. Quand l'agent Intégrateur détecte un besoin pour lequel il n'a pas les capacités nécessaires (ou pour lequel il ne trouve pas dans son SMA ces capacités), il utilise les fonctionnalités du module de recherche de *Gestionnaire des services* pour trouver des services Web qui pourraient satisfaire son besoin.

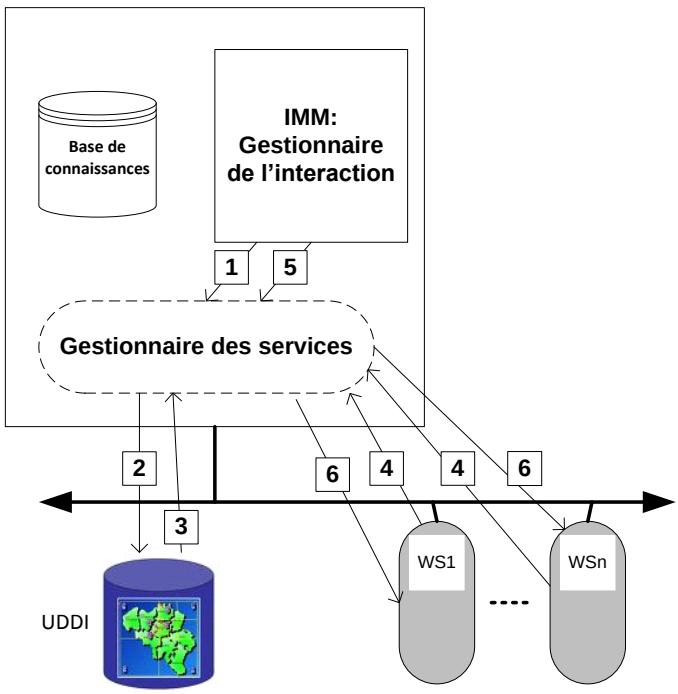

FIGURE 3.6 – *Communication Agent-Service Web*

2. Le module *Gestionnaire des services* interroge le registre UDDI.
3. Le module *Gestionnaire des services* obtient de la part du registre une liste de descriptions de services Web correspondant à ses critères de recherche.
4. Le module *Gestionnaire des services* commence l'invocation des services trouvés afin d'obtenir leurs différents attributs.
5. L'agent Intégrateur demande au module de *Gestionnaire des services* l'invocation d'un service Web, choisi parmi ceux dont la description lui a été envoyée dans l'étape précédente.
6. Le module *Gestionnaire des services* envoie en parallèle un message d'invocation au service choisi et des messages d'annulation aux autres services considérés.

3.6 Trace des interactions

Les agents dans notre SMA interagissent et se coordonnent en fonction des messages reçus et de ceux qu'ils se sont précédemment échangés. Pour ce faire, l'agent Intégrateur est muni d'une table d'historique. Une table d'historique regroupe les traces des interactions, notamment les précédents messages qu'il a échangés avec ses accointants (les agents participants). Une table d'historique est un ensemble d'enregistrements, cha-

cun étant dédié à une interaction qui représente l'ensemble de messages échangés pour la satisfaction d'un ensemble de besoins donnés, suivant le protocole d'interaction.

Un enregistrement d'une interaction est défini par le triplet (I_{id}, *Role*, *M*) où :

- I_{id} est l'identifiant de l'interaction.
- *Role* est le nom du rôle correspondant au protocole
- *M* est l'ensemble de messages échangés, i.e. envoyés et reçus, durant l'interaction.

3.7 Conclusion

Le développement des SMA pour l'intégration d'applications nécessite une bonne maîtrise des protocoles d'interaction et de tous les détails de leur utilisation. De ce fait, l'implémentation des protocoles d'interaction est complexe, de même pour l'implémentation des agents qui les utilisent.

Pour pallier ces problèmes, nous avons développé dans ce chapitre, une architecture basée-agent permettant l'intégration d'applications.

Hormis cet avantage crucial, notre architecture offre plusieurs autres avantages. C'est ainsi qu'elle est :

- flexible : car elle est facilement adaptable à d'autres applications grâce au concept du PI (il suffit de changer le fichier BPEL4WS qui encapsule le PI).
- interopérable : puisqu'elle supporte plusieurs modes de communication tels que la communication inter-agent et la communication agent-service Web.

Dans le chapitre suivant, nous illustrons l'application de notre approche avec une étude de cas. Nous décrivons tout d'abord l'utilisation des différentes étapes et modèles utilisés dans notre travail ainsi que l'architecture d'intégration proposée.

Etude de cas et implémentation

4

Sommaire

Nous venons de spécifier au long des précédents chapitres les concepts, les algorithmes et les structures de données permettant de concevoir une approche orientée interaction pour l'intégration des processus métiers. Dans le présent chapitre, nous présentons quelques aspects d'implémentation.

En effet, dans une première partie, nous proposons un scénario illustrant les différents aspects de notre approche. Dans une seconde partie, nous précisons la plate-forme d'implémentation JADE et comment les concepts de base utilisés dans notre architecture système peuvent être implémentés en utilisant cette plate-forme.

4.1 Etude de cas

L'évolution récente des infrastructures d'automatisation et de communication a permis au développement des applications e-business de prendre une autre dimension, particulièrement dans celles qui traitent de la coordination des processus dans les différentes entreprises. En fait, les applications e-business sont composées de plusieurs compagnies liées par des engagements afin de satisfaire les besoins de leurs clients. Les entreprises modernes qui sont de nature réparties englobent un grand nombre d'entités commerciales autonomes supportées par une variété de systèmes d'information hétérogènes.

Le paradigme de programmation orientée agent a été choisi pour implanter les différents mécanismes de notre architecture d'intégration. Dans le reste de ce chapitre, nous montrons l'application de notre approche avec une étude de cas, ainsi que la projection de l'architecture d'intégration que nous l'avons proposée sur un environnement opérationnel.

4.1.1 Présentation de l'étude de cas : Le système TeMS

Le TeMS (the Transportation e-Market System) est un système de gestion de transport des différents types. L'architecture de ce système est largement inspirée de nos travaux antérieurs [27][29]. Aussi, cette architecture a été implementée en 2006 dans le cadre d'un projet de fin d'étude pour l'obtention du diplôme d'ingénieur [28].

L'architecture de base de ce système (voir figure 4.1) est similaire à celle proposée par Karacapilidis et al. [94] dans le sens où elle comporte deux interfaces (avec les deux types d'utilisateurs du système : les clients et les fournisseurs des services) et un autre module qui gère les itinéraires. Notons que cette structure est à la base de toutes autres architectures de gestion des transports [5],[16].

Comme le montre la figure 4.1, l'architecture de TeMS illustre l'interaction entre trois parties : le CSD (*Customer Service Division*), le DD (*Design Division*) et le LD (*Legal Division*), où les deux premières parties sont des interfaces des systèmes, alors que la dernière est un service automatique.

Dans le reste de cette section nous présentons un éventuel scénario de ce système qui montre l'interaction entre les différentes parties. Ce scénario commence au moment où le rôle CSD envoie le message $request(ItineraryData)$. Une fois que le DD reçoit ce message, le service Web LD est invoqué pour la vérification de la légalité de l'itinéraire. Dans le cas d'un itinéraire légal, ce dernier est décomposé en en plusieurs sous-itinéraires.

Par la suite, le DD décide soit d'accepter la demande du client et propose alors le plan de l'itinéraire (l'envoi de message

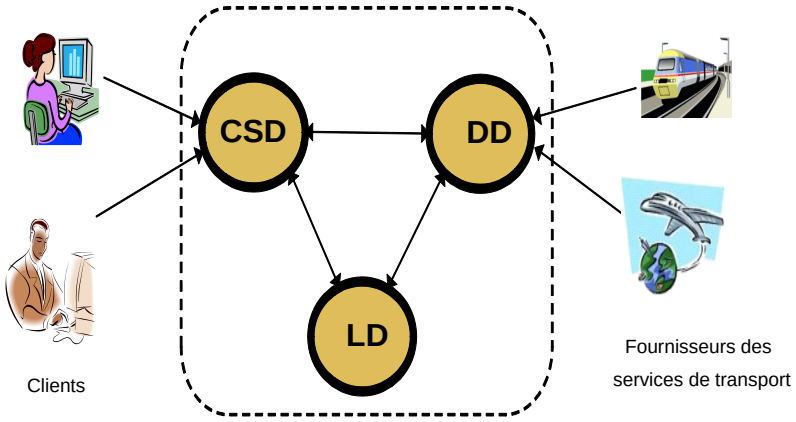

FIGURE 4.1 – *Système TeMS*

$propose(ItineraryPlan)$ au CSD) soit il refuse la demande du client. Dans notre modélisation, ce choix est représenté par un connecteur logique de type XOR, qui signifie qu'un seul message peut être envoyé.

Dans ce cas, le client a deux threads d'interaction qui représentent les messages entrants. Lorsque le CSD reçoit le message $propose(ItineraryPlan)$, il peut accepter ce plan de l'itinéraire ou déclarer un échec dans le protocole.

La section suivante a pour but d'appliquer, sur le scénario définit précédemment, notre démarche méthodologique décrite par le schéma de la figure 4.2. Dans cette dernière, le PI est modélisé à l'aide de diagramme d'interaction AUML. Ce modèle est d'abord transcrit en terme de specification BPEL4WS. Ceci permet de mettre en évidence le mécanisme général d'interaction entre les différents partenaires.

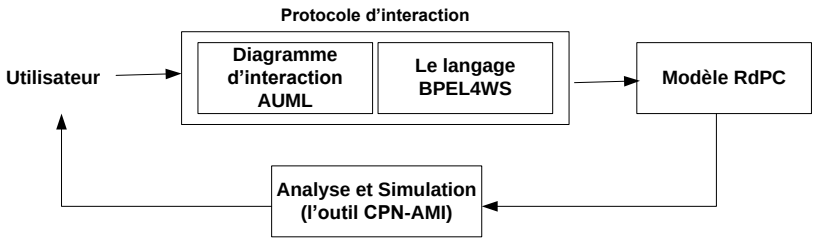

FIGURE 4.2 – *Démarche proposée*

Les specifications BPEL4WS obtenues sont ensuite transcrites en termes de RdPC. Ce modèle sert de base pour l'analyse par simulation, à l'aide de l'outil CPN-AMI.

4.1.2 Le diagramme d'interaction AUML : Une représentation graphique du scénario d'intégration

Un diagramme d'interaction peut être vu comme une séquence autorisée de messages entre agents et un ensemble de contraintes sur le contenu

de ces messages. Dans notre cas, ce diagramme permet de modéliser les interactions entre les différents processus du système. Un avantage majeur de cette représentation est qu'elle permet de cacher certains détails qui ne sont pas importants, ce qui le cas dans notre approche, où nous nous intéressons qu'aux actions de communication.

La figure 4.3-a montre un diagramme d'interaction AUML d'une partie de notre exemple. Nous rappelons que les processus privés ne sont pas définis par le protocole d'interaction, car le niveau d'abstraction choisi se restreint aux actions de communication observables entre les différents processus métiers.

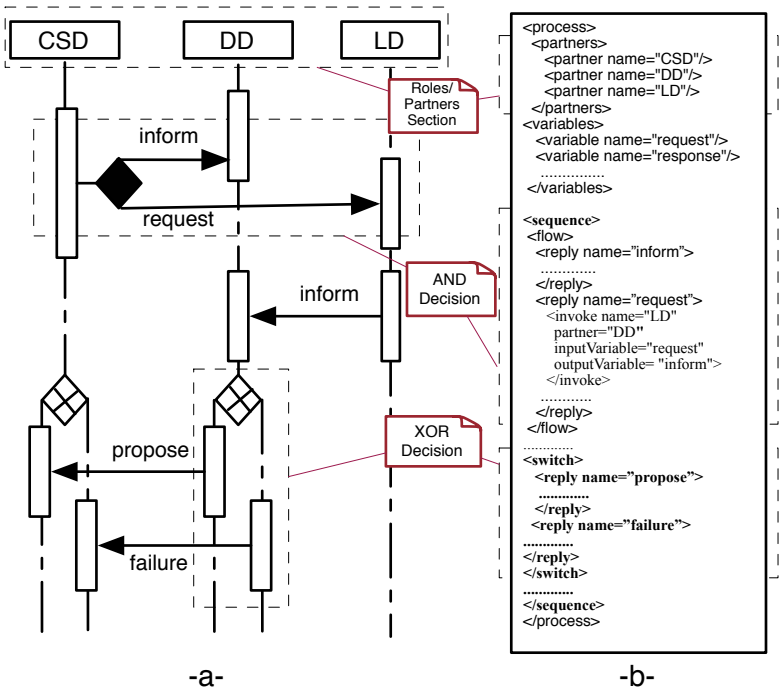

FIGURE 4.3 – *Protocole d'interaction comme AUML/BPEL4WS*

4.1.3 Le langage BPEL4WS : Une représentation textuelle du scénario d'intégration

Dans cette étape, le diagramme d'interaction AUML est d'abord transcrit en termes de BPEL4WS. Ceci permet de mettre en évidence le mécanisme général d'interaction entre les processus métiers.

En effet, le diagramme d'interaction d'AUML est traduit vers une specification BPEL4WS nommée par le nom du protocole. Cette spécification inclut huit ensembles abstraits qui sont représentés dans la table 1.7 de chapitre 2.

Nous commençons d'abord par la définition des partenaires de l'interaction (la partie $<$ *partners* $>$ de BPEL4WS), ensuite, nous passons à la

partie la plus importante a savoir la définition de l'interaction. Nous rappelons que l'interaction est une vue globale des actions observables des différents partenaires impliqués dans cette interaction.

Définition des partenaires

En AUML, les protocoles d'interaction sont présentés en général par des rôles. La partie "Rôle" du diagramme d'interaction est explicitement représentée par la section $<$ *partners* $>$ de BPEL4WS. Un partenaire est un processus (et/ou) un service Web évoluant dans une interaction. En effet, dans notre exemple, nous distinguons trois partenaires (voir Fig. 4.4) : le processus CSD (customer service division), le service DD (design division) et le service Web LD (legal division).

```
<partners>
        <partner name="CSD"/>
        <partner name="DD"/>
        <partner name="LD"/>
</partners>
```

FIGURE 4.4 – *Définition des partenaires de l'interaction en BPEL4WS*

Définition de l'interaction

L'interaction est une vue globale des actions observables des différents partenaires impliqués dans cette interaction. Ces actions correspondent à l'échange de messages.

En fait, cette étape consiste à définir l'ensemble des messages échangés durant l'interaction ainsi que l'association de chaque message d'un partenaire à son partenaire cible.

Par conséquent, nous exploitons la partie $<$ *variable* $>$ de BPEL4WS pour définir touts les messages échangés dans le diagramme d'interaction AUML (comme le montre la Fig. 4.5).

Ensuite, la description du processus abstrait de l'interaction, est décrite dans le langage de description comportementale BPEL4WS en appliquant les règles de correspondances entre AUML et BPEL4WS définies dans le chapitre 2. La spécification BPEL4WS obtenue du comportement observable de l'interaction des trois partenaires, est représentée par la Fig. 4.3-b.

La combinaison AUML/BPEL4WS permet une description très claire et détaillée du scénario d'intégration. Néanmoins, bien que très perfor-

```
<variables>
        <variable name="request"/>
        <variable name="propose"/>
        <variable name="inform"/>
        <variable name="failure"/>
</variables>
```

FIGURE 4.5 – *Définition des messages en BPEL4WS*

mante, cette combinaison n'offre aucune structure formelle pour vérifier, valider et évaluer les performances du système avant les phases d'implémentation.

Par conséquent, il apparaît nécessaire de disposer d'un ensemble d'outils formels capables de prendre en compte ces aspects de validation. L'approche proposée dans ce travail, est basée sur l'utilisation des RdPC. Ce formalisme offre la possibilité de décrire le système d'une façon claire et réaliste. Par ailleurs, chaque jeton de couleur différente représente des valeurs de données arbitraires. Cette extension augmente la puissance de modélisation. Le franchissement des transitions dépend de la disponibilité des jetons avec les couleurs appropriées. Enfin, les modèles obtenus sont directement exécutables permettant l'analyse et la validation des protocoles à l'aide du l'outil CPN-AMI.

4.1.4 Génération de RdPC

Avant de créer le RdPC, il est nécessaire de définir la gestion des variables en utilisant des types de données et des domaines de couleurs. Un domaine de couleur est associé aux places et aux transitions dans le modèle de réseau de Petri pour représenter l'information exigée afin de determiner l'état du système.

Dans notre travail, nous considérons les domaines de couleurs et les variables suivantes :

Colour sets :

$Communicative\ Act = \textbf{with}\ inform|cfp|propose|\ldots;$
$Role = string\ \textbf{with}\ a\ldots z;$
$Content = string\ \textbf{with}\ a\ldots z;$
$Bool = \textbf{with}\ true|false;$
$MSG = record$
$\quad s, r : Role;$
$\quad CA : Communicative\ Act;$
$\quad C : Content;$

Variables :
msg, *msg*1, *msg*2 : *MSG*;
x : *Bool*;

Comme nous l'avons présenté dans le chapitre 2 de ce manuscrit, le domaine de couleur *MSG* décrit le type de messages dans une communication inter-processus. Dans ce cas *MSG* est un enregistrement $< s, r, ca, c >$, où les éléments *s* et *r* déterminent l'expéditeur et le récepteur du message correspondant. Ces éléments ont le type *Role*, qui est utilisé pour identifier les partenaires de l'interaction. Le type "Communicative Act" et *Contents* représentent respectivement les actes communicatifs FIPA-ACL et le contenu du message correspondant.

Dans cette section, nous utilisons l'algorithme IP2CPN développé dans le chapitre 2 afin de générer le RdPC correspondant au diagramme d'interaction AUML présenté dans la figure 4.3.

Comme l'indique la règle 1 (voir section 3.5 de chapitre 2), nous utilisons une approche par composition de réseaux, dans laquelle chaque rôle de l'interaction est représenté par les séquencements de places et de transitions appartenant à ce rôle. Le réseau est donc constitué d'un sous-réseau pour chacun des rôles qui participe à l'interaction.

Par conséquent, l'algorithme commence avec la création de trois places Rôles (*RP*), initialement marquées. Selon la règle 1, l'algorithme permet d'associer une place pour chaque intervenant dans l'interaction (les lignes 3 et 4). La ligne 5 permet de mettre à jour le RdPC avec la variable *RP* .

Dans la première itération de la boucle principale (ligne 7), la variable *curr* prend le premier message du protocole d'interaction (*curr* $\leftarrow$ <$CSD, DD, inform, A > AND < CSD, LD, request, S$>).

Il s'agit du cas où plusieurs messages sont envoyés en même temps. Dans ce cas, la règle 4.3 est appliquée pour créer une transition sans garde et deux places de synchronisation. Dans notre exemple, l'algorithme crée les places du réseau, qui sont associées à la variable *curr* , c'est-à-dire les places correspondent aux messages "$inform$" et "*request*" (ligne 8) et deux places intermédiaires dans le sous-réseaux DD et LD respectivement (la fonction $CreateIntermediatePlacee()$ à la ligne 9) .

Ces places (voir les RdPC de la figure 4.6) sont connectées avec des arcs et des transitions (en appliquant la règle 4.3). La variable *MP* n'est pas une place finale (le CSD est en attente d'une réponse de DD), donc cette place est connectée par des transitions et des arcs avec les fonctions $CreateTransitions()$ et $CreateArcs()$ (lignes 10, 11, 12). Ensuite, nous ajoutons les couleurs des jetons à la structure de RdPC, en utilisant la fonction $FixColor()$ (la ligne 16).

Dans la deuxième itération, la variable *curr* est fixée à <LD, DD, in-

form, A>. L'algorithme crée une place pour le message *inform* (la fonction *CreateMessagePlace*() à la ligne 8) et deux places dans les sous-réseaux LD et DD respectivement (la fonction *CreateIntermediatePlace*() à la ligne 9). Ces places sont connectées à l'aide du bloc de RdPC précédemment décrits (la règle 2).

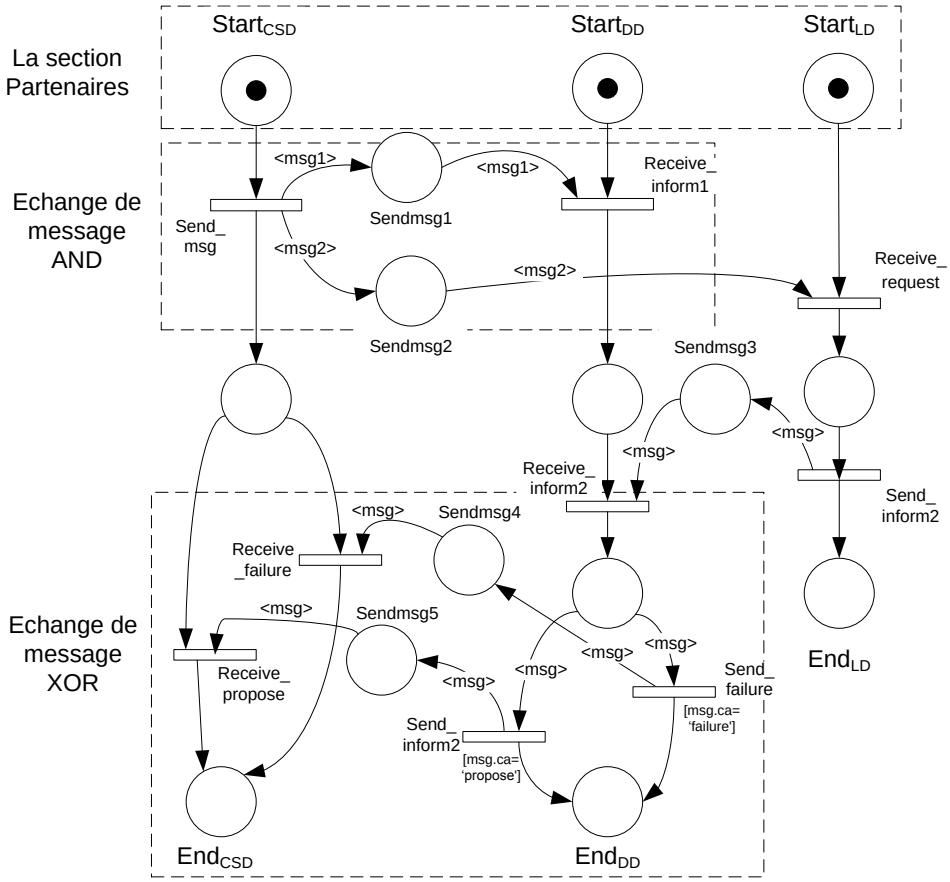

FIGURE 4.6 – *Le RdPC de protocole d'interaction*

Dans la dernière itération, *curr* est fixé à <DD, CSD, propose, a> XOR <DD, CSD, failure, a>. Dans ce cas, le DD peut envoyer soit un *failure* ou un *propose*, et donc les places pour ces deux messages sont créés en utilisant le bloc XOR-décision indiqué dans la règle 4.1. Ensuite, deux places, qui correspondent aux résultats des messages sont crées. Ces places sont connectées à l'aide du bloc XOR-décision décrit dans la règle 2. Cette règle implique la création d'une garde de contrôle sur les franchissements des transitions qui correspondent au type de message (ce qui est représenté par une couleur dans le RdPC).

Dans cette itération, nous notons que la variable MP (correspond au message *failure*) est une place finale. Donc il n'y a pas de transition ou d'arcs sortant crées à partir de cette place.

4.1.5 Analyse du modèle réseau de Petri

Les simulations de la dynamique du réseau de Petri nous ont permis de valider chaque module pendant son élaboration, en vérifiant s'il présente le comportement attendu. Il existe deux approches principales d'analyse d'un réseau de Petri : l'identification des propriétés structurelles et l'analyse par model checking du graphe des marquages accessibles [71]. L'analyse structurelle est fondée sur les propriétés algébriques du réseau, tandis que le graphe des marquages accessibles est un graphe orienté dont les noeuds correspondent aux états accessibles du système et les arcs aux transitions franchissables entre ces états. Après la phase de débogage, nous avons vérifié la cohérence du système grâce aux propriétés d'invariants structurels.

Les spécifications structurelles peuvent s'adapter à la plupart des modèles et sont généralement constituées de « bonnes propriétés » que l'on cherche à retrouver par des méthodes d'analyse parfois coûteuses. Au contraire, les spécifications comportementales sont spécifiques au système à modéliser et constituent le coeur du modèle.

Cette étude ayant davantage un intérêt qualitatif. Elle consiste à construire le graphe global de l'interaction où chaque état correspond à un état de la communication.

Les états et la structure de ce graphe sont ensuite examinés à la lumière d'un certain nombre de propriétés telles que l'interblocage et la réception non spécifiée. La vérification des ces propriétés permettent de valider le protocole d'interaction mis en oeuvre.

Dans notre travail, nous avons utilisé l'environnement logiciel CPN-AMI [1] développé au LIP6 [2] . Cet environnement est dédié à la théorie des réseaux de Petri colorés et supporte la spécification, la validation, la vérification formelle et le prototypage conforme d'applications coopératives. Ses outils permettent de vérifier des formules de logique temporelle (model checking), de calculer des propriétés structurelles (invariants, verrous, trappes), de simuler, de déboguer et d'engendrer du code [74]. L'accès aux outils se fait à travers l'interface utilisateur *Coloane* [22] qui est un éditeur de graphes génériques.

Coloane est un environnement graphique dédié à la production de modèles et à l'invocation de services de l'environnement CPN-AMI. Il poursuit donc deux objectifs :

1. Fournir les outils nécessaires au développeur pour qu'il puisse modéliser un système en utilisant les formalismes mis à sa disposition (potentiellement plusieurs).

1. http ://www.lip6.fr/cpn-ami
2. Laboratoire d'Informatique de Paris 6

2. Permettre la connexion à l'ensemble des outils de CPN-AMI en respectant le protocole d'échange, l'invocation des services disponibles sur la plate-forme et être capable d'exploiter les résultats de celles-ci.

Les états de ce graphe sont ensuite examinés à la lumière d'un certain nombre de propriétés. En fait, dans notre cas, nous nous intéressons à deux types de propriétés :

– des propriétés générales que l'on retrouve dans tous les protocoles comme les problèmes d'interblocage, de réception non spécifiée,etc. et
– des propriétés fonctionnelles qui sont dépendantes de la fonction du protocole et qui sont décrites dans le cahier des charges lors de la phase d'analyse.

Propriétés générales

Les propriétés générales représentent les conditions de base de la validité des spécifications des protocoles. Ces propriétés peuvent être résumées comme suit :

1. **Absence d'interblocage :** Comme nous l'avons présenté dans la section 6 de chapitre 2, le RdP vérifie cette propriété si et seulement si touts les rôles impliqués dans une interaction atteignent leurs états de fin respectifs. Plus précisément, soit END_r la place terminale pour chaque rôle r impliqué dans l'interaction et soit *endsta* l'ensemble des états terminaux du graphe des états accessibles. La propriété de l'interblocage est vérifiée si : pour chaque $r \in$ {CSD,LD,DD} alors $End_r \in endsta$. Cela est exprimé en CTL comme suite :

$$\forall r \in R, End_r \in endsta, AF(card(End_r) > 0)$$

2. **Vivacité :** Tous les états sont atteignables à partir de l'état initial et la consommation de tous les messages du protocole est assurée. Dans ce cas, il suffit de vérifier que les jetons des places de synchronisation (les places $Sendmsg_i$ dans notre exemple) sont consommés. Donc, dans notre exemple on distingue 5 requêtes CTL (autant que le nombre des places de synchronisation) qui seront exécutées sur le graphe des états accessibles.

$$\forall i \in [1,5], AG(IfThen(card(Sendmsg_i) > 0, \\ AF(card(Sendmsg_i) == 0)))$$

Propriétés spécifiques aux protocoles d'interaction

Ces propriétés dépendent de domaine d'application et sont décrites dans le cahier des charges lors de phase d'analyse. En effet, les propriétés suivantes sont basées sur notre exemple.

Les propriétés spécifiques au protocoles d'interaction peuvent être exprimées avec la sémantique de logique temporelle (la logique CTL dans notre cas).

Par exemple, le protocole doit assurer que, après la demande CSD (l'envoi de message *request*) , il faut éventuellement recevoir un résultat. Dans notre exemple, la CSD devrait recevoir soit un message *propose* soit un message *reject* (voir Figure 4.3). La formule CTL suivante permet de capturer cette propriété où les places $Sendmsg_i$ correspondent à l'échange de messages (voir section 4.1 du chapitre 2).

$$AG(IfThen(card(Sendmsg_1 : (field[2] == request)) > 0,\\ AF(card(Sendmsg_4 : (field[2] == propose)) > 0\, or\\ card(Sendmsg_5 : (field[2] == failure)) > 0)))$$

Du point de vue de la DD, il tient à assurer que le service LD envoie l'état de la légalité du service proposé. Pour vérifier cette propriété, la formule suivante est utilisée.

$$AG(IfThen(card(Sendmsg_2 : (field[2] == request)) > 0,\\ AF(card(Sendmsg_3 : (field[2] == inform)) > 0)))$$

4.2 Implantation de l'architecture proposée

Un système à base d'agents implantés sur différentes machines peut être considéré comme une application répartie dans laquelle les entités sont des agents. Cette caractéristique exige que ces agents soient déployés dans un environnement qui supporte leur exécution.

Comme nous avons mentionné dans la section 2 de chapitre 3, plusieurs architectures basées sur l'agent pour l'intégration d'applications ont été proposées. Cependant, très peu de corrélation est établie entre les architectures logicielles proposées et ses projection sur un environnement opérationnel. Il n'existe à l'heure actuelle que peu de tentatives visant à concilier ces deux aspects.

Pour faire face à ce point de rupture, nous préconisons que la construction d'un système à base d'agents doit correspondre à un processus qui prenne en compte la définition du système aussi bien que son implantation dans l'environnement cible qui supportera son fonctionnement.

De point de vue de mise en oeuvre, nous avons utilisé une plate-forme basée-agent pour implémenter les différents agents et concepts. Le choix d'une telle plate-forme est basé sur plusieurs critères comme :

- La nature des agents ;
- Le type de communication employée ;
- Le protocole d'interaction ;
- Les mécanismes de sécurité ;
- Le degré de standardisation établi dans cette plate-forme.

Cette plate-forme va nous permettre d'implémenter les différents types d'agents de notre architecture et de simuler l'interaction entre l'agent Intégrateur et les agents d'entreprise durant la phase d'intégration.

Plusieurs plates-formes sont fournies comme des logiciels libres. Elles ont été utilisées dans le développement de plusieurs applications. Parmi elles ont peut mentionner : JADE [19], ZEUS [13] et MADKIT [95] pour les agents cognitifs, et SWARMS [123] pour les agents réactifs. Il faut noter que cette liste n'est pas unique, et qu'il y a aussi d'autres plates-formes qui ont été utilisées avec beaucoup de succès pour bâtir diverses applications.

4.2.1 La plate-forme JADE

JADE (Java Agent Development Framework) est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie). Elle fournit un environnement de développement et d'exécution des systèmes multi-agents, compatible avec les standards FIPA [19]. JADE comprend deux composantes de base : la plate-forme d'agents compatible FIPA et un package logiciel pour le développement des agents Java. Elle fournit les facilités suivantes :

Une plate-forme agent répartie : la plate-forme peut être distribuée (partagée) entre plusieurs hôtes (hosts) connectés via RMI de Java, de telle façon qu'une seule application Java, et par conséquence une seule " Machine Virtuelle Java " est exécutée sur chaque hôte.

Un certain nombre de DF (Facilitateurs d'Annuaire) compatibles FIPA : qui peuvent être activés quand on lance la plate-forme pour exécuter les applications multi-domaines.

interface utilisateur graphique (GUI) : pour gérer plusieurs agents et conteneurs d'agents d'un hôte éloigné.

Outils de Déboguage : pour faciliter la mise au point d'applications.

Un support d'exécution : pour les activités multiples, parallèles et concourantes des agents via le modèle de comportement (behaviour)

Un transport efficace des messages ACL : à l'intérieur de la même plate-forme. Cette conversion est transparente aux programmeurs intéressés uniquement aux objets Java.

Une bibliothèque de protocoles : compatibles aux standards FIPA et prêts à être employés pour régir l'interaction inter-agent.

A l'image de ces avantages, il est clair que JADE est la plate-forme le plus proche de nos critères. En effet, dans notre cas la plate-forme utilisée est JADE et le but de cette section est d'expliquer la correspondance qui peut exister entre les concepts de base proposés par notre approche (capacités, plans, interactions, etc.) et ceux fournis par la plate-forme JADE pour l'implantation de système [27].

4.2.2 Utilisation de JADE pour le développement de l'architecture proposée

JADE utilise l'abstraction de comportement pour modéliser les tâches qu'un agent peut exécuter et les agents instancient leurs comportements selon leurs besoins et leurs capacités.

De point de vue de la programmation concurrente, un agent est un objet actif, ayant un thread de contrôle. JADE utilise un modèle de programmation concurrente " un thread-par-agent " au lieu de modèle " un thread-par-comportement " pour éviter une augmentation du nombre de threads d'exécution exigés sur la plate-forme d'agents. Ceci signifie que, pendant que les agents différents s'exécutent dans un environnement multi-threads préemptive, deux comportements d'un même agent sont planifiés d'une façon coopérative.

Du point de vue implantation, un agent dans JADE est une instance de la classe Java définie par le programmeur. Cette classe elle-même est une extension de la classe Agent de base (incluse dans *jade.core*). Cela implique l'héritage de l'ensemble de méthodes de base pour implémenter le comportement personnalisé de l'agent. L'implémentation d'un agent est multitâche, où les tâches (appelées comportements) sont exécutées en concurrence. Chaque fonctionnalité fournie par un agent doit être implémentée en un ou plusieurs comportements. La Figure 4.7 présente le code Java, pour la définition de la classe *IntegratorAgent* et la classe *EntrepriseAgent*. Cela est fait par l'extension de la classe de base *Agent* définie dans JADE.

Le module de communication présenté dans notre architecture est dérivée de la classe *CyclicBehaviour* de la plate-forme Jade, elle s'exécutera donc continuellement. Une fois qu'un message est arrivé, le module de communication interprète les informations obtenues. Ensuite, elle déclenche le module d'execution et retourne à l'état initial (attendre les mes-

```
public class Integrator extends Agent {
      protected void setup() {
addBehaviour(new SimpleBehaviour(this)
{

                  // Traitement ...
      }
}
```

```
public class EntrepriseAgent extends
Agent {
  class recevoir extends
SimpleBehaviour {

      //Traitement ...
  }
  public recevoir(Agent a)
    {
      super(a);
    }
  protected void setup() {
    recevoir mybehaviour = new
recevoir(this);
    addBehaviour(mybehaviour);
  }}
```

FIGURE 4.7 – *Extension de la classe Agent*

sages). On montre le code de comportement de module de communication ci-dessous (4.8).

```
import java.util.*;
import jade.core.*;
import jade.core.behaviour.*;
import jade.lang.acl.ACLMessage.*;
class PeceptionInterface extends CyclicBehaviour {
 public PeceptionInterface (Agent a) {super(a);}
 Vector goals = new Vector();
 public void action() {
  // wait for message
  ACLMessage received = myAgent.receive();
  if(ACLMessage == null) {block();}
  else {
   // message interpretation
   .....
   // Start PlanRetrieval Behaviour
   addBehaviour(new PlanRetrieval(this,goals));
    }
   }
  }
```

FIGURE 4.8 – *Spécification partielle de module de communication exprimée avec JADE*

Le module IMM (Interaction Manager Module) de l'agent Intégrateur est dérivé de la classe *SimpleBehaviour*. Ce module prend comme paramètres d'entrée une spécification BPEL4WS et construit le protocole d'interaction adéquat. Dans ce contexte, un protocole d'interaction est un comportement de type *FSMBehaviour*. Un comportement de type *FSMBehaviour* est un comportement composé (*CompositeBehaviour*) qui exécute ses sous-comportements selon une machine à états finie définie

par l'utilisateur. Chaque sous-comportement représente l'activité à être exécuté dans un état du comportement *FSMBehaviour* et l'utilisateur peut définir les transitions entre ces états. En effet, chaque sous-comportement est dédié à une interaction qui représente l'ensemble des messages échangés pour la satisfaction d'un ensemble de besoins donnés.

4.2.3 Intégration de XML dans FIPA-ACL supportant la communication inter-agents

La réussite de notre architecture passe également par une communication efficace tout au long de son déroulement. Le mode de communication que nous avons adopté est celui par envoi de messages. En effet, analyser des données, identifier des informations, maintenir à jour des connaissances, communiquer les synthèses et les prises de décision des agents hétérogènes par leur fonction et par leur domaine d'activités, travailler en coopération, nécessite un mécanisme de communication sans faille.

Notre choix de sélection du langage de communication est basé sur celui qui favorise le plus l'interopérabilité. Notre préoccupation est de définir un langage de communication pour faire interopérer des agents, décrire les actions que peut accomplir chaque agent, et comprendre la sémantique de chaque tâche à accomplir.

L'enjeu principal de FIPA est la standardisation dans un but d'interopérabilité les systèmes à base d'agents logiciels hétérogènes. Parmi ces standards de communication, le langage FIPA-ACL [44], devenu le standard incontournable pour l'interaction des agents et plus riche sémantiquement que son rival KQML [127]. Dans notre système l'intérêt principal de la standardisation est de faire interopérer la société d'agents, d'où la motivation de notre choix pour le langage FIPA-ACL.

Dans le langage FIPA-ACL, aucun langage spécifique pour la description des contenus des messages n'est imposé. Dans notre travail, le langage XML est utilisé pour la description, la spécification et l'interprétation du contenu des messages (Content) échangés entre les différents agents. Donc, les messages échangés entre les agents sont décrits dans FIPA-ACL et XML.

L'exemple de la fig. 4.9 illustre l'interaction entre l'agent Intégrateur et un agent d'entreprise (1). L'agent Intégrateur annonce le sous but (les sous-itinéraires, spécifié dans la partie "content" du message selon la structure décrite auparavant).

4.2.4 Invocation des processus métiers durant la phase d'exécution

Afin de pouvoir fournir des services ou des produits aux participants durant le processus public (le processus de coopération entre plusieurs

```
(cfp
:sender Integrator-Agent
:receiver Enterprise-Agent1,...
:language XML
:protocol IP-1
:content ( <Sub_Itinerary SI_id = 'CDZ-ORL'>
               <List_Att>
               <detart_day> 01102009 </depart_day>
               <return_day> 01102009 </return_day>
               <List_Att>
                 <Attibut Att_name = 'price'>
                    <Dom Val_min='140' Val_max='160'/>
                 </Attribut>

                 <!-- ... etc. -->
               </List_Att>
           </Sub_Itinerary>
:reply-with offre_Itinerary
)
```

FIGURE 4.9 – *Exemple d'un message ACL/XML*

entreprises), les processus métiers sont exécutés localement à chaque entreprise sous-jascente à un agent participant.

Le module de connaissances individuelles de l'agent d'entreprise (voir la section 3.3 de chapitre 3), contient les informations sur les ressources internes de l'entreprise (application, usagers, sources de connaissances, etc.) qui permettent la réalisation des tâches locales assignées à cette entreprise. Ce module a le rôle de réaliser la correspondance entre le sous but (la tâche) assigné à l'agent et les ressources internes de l'entreprise aptes à l'achèvement de cette tâche. La reception d'un message (spécifié avec ACL/XML) consiste à lancer l'exécution du processus métier concerné.

4.2.5 Outils techniques utilisés

Les outils techniques que nous avons utilisé pour la publication, la découverte et le déploiement des services sont :

- Tomcat : qui est un serveur d'application pour l'utilisation de servelts, de fichiers .jsp et .java.
- jUDDI : implémentation open source de l'annuaire UDDI. jUDDI gère le transport des données XML grâce à Axis (qui est considéré principalement comme un moteur SOAP). Il tourne par ailleurs sur le serveur Tomcat.
- MySQL : représente la base de données supportant l'annuaire jUDDI.
- UDDI4J : est une API Java permettant d'interagir avec l'annuaire

jUDDI, c'est-à-dire de s'y connecter, de publier et de rechercher des services.

4.3 Discussion

Dans les sections précédentes, nous nous sommes intéressés à l'implementation de notre architecture d'intégration. Cette architecture comporte un ensemble d'agents pouvant être utilisés pour les applications opérant sur des sources d'information hétérogènes placées dans un environnement ouvert et dynamique (comme par exemple l'internet). Cette exigence de "réutilisabilité", nous a été dictée par le fait que la complexité engendrée par un tel système est parfois supérieure aux grands projets informatiques. De ce fait, le coût de production de ce genre de systèmes est énorme en personne-mois tant au niveau de la conception, que de la réalisation et de la vérification. La réutilisabilité est un moyen pour réduire la charge de travail et par le fait même, le coût de production et c'est ce que nous avons choisi.

Hormis cet avantage crucial, notre architecture offre plusieurs autres intérêts. En effet, elle est :

- portable : car elle est programmée en Java ;
- flexible : facilement adaptable à d'autres applications grâce au concept du protocole d'interaction (il suffit de changer le fichier BPEL4WS qui encapsule le protocole d'interaction).
- interopérable : puisqu'elle support plusieurs modes de communication tels que la communication inter-agent et la communication agent-service Web.

Les premiers essais effectués ont montré que l'idée de créer un agent intermédiaire pour exécuter les protocoles d'interaction offre beaucoup d'avantages, notamment, la réutilisation et la maintenance des protocoles d'interaction. En plus, l'approche proposée évite d'encombrer les agents participants avec les règles d'interaction.

Bien que notre architecture offre plusieurs avantages, il y a toujours place pour son amélioration. C'est ainsi que nous prévoyons d'inclure deux nouveaux mécanismes : un mécanisme pour la gestion des nouveaux agents entrant dans le système et un autre mécanisme pour détecter les incohérences d'un protocole d'interaction pendant son exécution.

4.4 Conclusion

Dans la première partie de ce chapitre, nous avons présenté les étapes de formalisation et de vérification des PI à travers un scénario illustrant différents aspects complémentaires à ces derniers.

Dans la seconde partie de ce chapitre, nous avons défini l'architecture globale d'intégration d'applications, en reliant nos précédentes spécifications au contexte et aux technologies du Web. Nous avons ainsi décrit les phases de publication et d'invocation des processus métiers et des services web.

Pour implémenter cette architecture, nous nous sommes basés sur des standards. Pour cela, nous avons utilisé les technologies FIPA-ACL et XML pour représenter les informations échangées entre les agents durant l'interaction. L'architecture proposée a été implantée en utilisant la plate-forme Jade. Cette plate-forme permet de simplifier le développement des systèmes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA.

Conclusion générale

5

L'intégration d'applications d'entreprises impose le traitement de nombreux mécanismes, tels que la communication et la coordination de différents processus. Ce traitement pose de nombreux problèmes de validation particulièrement en termes d'échange de messages entres les différents processus.

Dans cette thèse nous avons traité le problème de l'intégration d'applications d'entreprises (IAE). Cette intégration consiste à coordonner des processus métiers (issus de différentes organisations) afin d'atteindre un objectif commun. Notre objectif était de spécifier des propriétés et des protocoles d'interaction SMA (Système Multi-Agents), vérifier leurs comportements, et de les adapter pour intégrer plusieurs applications d'entreprises.

5.1 Contributions

Dans ce contexte, plusieurs contributions ont été développées :

La modélisation de scénario d'intégration avec le concept de protocoles d'interaction

Ce travail propose une approche orientée **interaction** (observabilité des actions de communication) pour l'IAE[26]. L'intérêt de notre approche est d'appréhender les interactions dans les SMA à travers des méthodes pour la modélisation, l'analyse et la vérification.

Ainsi, nous avons utilisé la combinaison AUML/BPEL4WS pour la modélisation des interactions entre les processus métiers. AUML sert de représenter les protocoles d'interaction (PI) d'une manière graphique. Le langage BPEL4WS décrit ces interactions à travers un plan en spécifiant les flots de contrôle entre les services partenaires, ainsi que les dépendances des données entre processus métiers.

La formalisation des protocoles d'interaction avec les réseaux de Petri colorés

Nous avons apporté une sémantique opérationnelle pour les modèles des PI définis dans notre approche [25]. Cette sémantique est basée sur des règles de transformation garantissant la cohérence du système. Le modèle formel retenu pour la représentation du comportement observable des processus métiers est les réseaux de Petri colorés. Nous avons aussi développé l'outil *IP2CPN* pour la génération automatique des réseaux de Petri colorés.

Nous avons étendu l'approche proposée à la vérification des PI. Pour cela, nous avons utilisé l'environnement logiciel CPN-AMI. Deux types de propriétés sont vérifiées à l'aide de cet environnement. Le premier concerne des propriétés générales que l'on retrouve dans tous les protocoles comme les problèmes d'interblocage et de réception non spécifiée. Le second porte sur des propriétés fonctionnelles qui dépendent de la fonction du protocole et qui sont décrites dans le cahier des charges lors de la phase d'analyse.

La gestion de l'interaction par le SMA

L'architecture du système qui en découle de l'approche proposée est basée sur le concept d'agent[24]. Cette architecture repose sur deux éléments clés : un framework modulaire d'agents interactifs et une bibliothèque de protocoles réutilisables.

Notre architecture possède des fonctionnalités qui prennent en charge les mécanismes de communication et de coordination tels que la communication inter-agent et la communication agent-sercices Web.

L'implémentation de cette architecture est basée sur la plate-forme Jade. Cette plate-forme permet de simplifier le développement des systèmes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA. Aussi, nous avons utilisé les standards FIPA-ACL et XML pour représenter les informations échangées entre les agents durant l'interaction.

5.2 Perspectives

Les travaux proposés dans cette thèse nous permettent d'ouvrir plusieurs perspectives à court et à moyen terme.

Perspectives à court terme

La proposition d'une ontologie de protocoles d'interaction. Il s'agit de définir les concepts communs à ces protocoles, leurs liens et les

axiomes exprimant des contraintes portant sur ces concepts. Cette ontologie offre une sémantique claire aux protocoles. Cela permet des interactions complexes et indépendantes du domaine dapplication.

Optimisation de l'échange de messages. Au moment de l'interaction des agents, le flux de message produit peut être réduit par l'exploitation des traces des interactions précédemment effectuées. Le but est de diminuer la complexité de communication entre les agents.

L'intégration des protocoles. Si un agent requiert des fonctionnalités, et qu'aucun protocole d'interaction n'est apte à les fournir, il devrait être possible de combiner des protocoles existants afin de répondre aux besoins de cet agent. A ce stade, il est nécessaire de connaître la structure interne de chaque protocole pour pouvoir le composer correctement. Une étape de vérification est alors nécessaire pour assurer que l'intégration des protocoles est correcte.

Perspectives à moyen terme

La vérification automatique des protocoles d'interaction. L'approche que nous avons proposé ne traite pas le cas de changement des protocoles. Ainsi, pour chaque changement d'un protocole, il faut refaire l'étape de vérification (pour assurer que le protocole fonctionne correctement). Donc, il serait intéressant d'étudier les possibilités d'automatiser la vérification des protocoles et encore de détecter les incohérences d'un protocole d'interaction pendant son exécution.

Réutilisation des protocoles d'interaction. Il serait intéressant d'étudier la mise en place d'un mécanisme d'apprentissage utilisant les résultats des observations afin d'améliorer les modes d'interaction entre les agents mis en oeuvre. Ce mécanisme permet d'enrichir les connaissances d'un agent par de nouveaux types d'interactions.

Bibliographie

[1] A. Berger and S. Pesty. Towards a conversational language for artificial agents in mixed community. In *Proc. of the 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'05)*, pages 31–40, 2005. (Cité page 26.)

[2] A. Fallah-Seghrouchni, S. Haddad, and H. Mazouzi. Protocol Engineering for Multi-agent Interaction. In *MAAMAW'99 : Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 89–101, London, UK, 1999. Springer-Verlag. (Cité pages 37, 45 et 47.)

[3] A. Gustavo, F. Casati, H. Kuno, and V. Machiraju. *Web Services. Concepts, Architectures and Applications*. Springer-Verlag, Berlin Heidelberg, 2004. (Cité page 3.)

[4] A. Hamez, L. Hillah, F. Kordon, A. Linard, E. Paviot-Adet, X. Renault and Y. Thierry-Mieg. New features in CPN-AMI 3 : focusing on the analysis of complex distributed systems. In *Sixth International Conference on Application of Concurrency to System Design (ACSD 2006)*, pages 273–275. IEEE Computer Society, 2006. (Cité pages 7 et 64.)

[5] A. Jeffrey, S. Goutam, V. Manikonda, B. Betty and B. Victor. A multiagent approach to cooperative traffic management and route guidance. *Transportation Research Part B : Methodological*, 39(4) :297–318, May 2005. (Cité page 87.)

[6] A. Licci, E. Denti, and A. Omicini. Agent coordination infrastructures for virtual enterprises and workflow management. In M. Klugch and F. Zombonelli (Eds), editors, *5th Int. Workshop of Cooperative Information Agent V (CIA 2001)*, pages 235–246, Italy, 2001. , LNAI, Springer. (Cité page 72.)

[7] M. Vidal A. Paul and H. Verhagen. Adaptive Workflow = Web Services + Agents. In *Proceedings of the International Conference on Web Services : ICWS'03*, pages 131–137, 2003. (Cité pages 18 et 45.)

[8] A. Perez, and V. Benjamins. Overview of Knowledge Sharing and Reuse Components : Ontologies and Problem-Solving Methods. pages 1–1, 1999. (Cité page 72.)

[9] J. Austin. *How to Do Things with Words*. Oxford Press, 1962. (Cité page 27.)

[10] B. Bauer, F. Bergenti, P. Massonet, and J. Odell. Agents and the UML : A Unified Notation for Agents and Multi-agent Systems. In *Agent-Oriented Software Engineering II, Second International Workshop, AOSE'01*, volume 2222 of *LNCS*, pages 148–150. Springer, 2001. (Cité pages 7, 31, 44, 47 et 52.)

[11] B. Bauer, J. Muller, and J. Odell. Agent UML : A Formalism for Specifying Multiagent Software Systems. In *Agent-Oriented Software Engineering, First International Workshop, AOSE'00*, volume 1957 of *LNCS*, pages 91–104. Springer, 2000. (Cité pages 32 et 47.)

[12] J. Barjis. Collaborative, Participative and Interactive Enterprise Modeling. In *ICEIS'09*, volume 24 of *Lecture Notes in Business Information Processing*, pages 651–662. Springer, 2009. (Cité page 16.)

[13] C. Collis, and L. Lyndon. Building Electronic Marketplaces with the ZEUS Agent Tool-kit. *Lecture Notes in Computer Science*, 1571 :1–19, 1999. (Cité page 97.)

[14] C. Girault, and R. Valk. *Petri Nets for Systems Engineering, A Guide to Modeling, Verification, and Applications*. Springer Verlag, July 2002. (Cité pages 7, 36, 44 et 56.)

[15] C. Hanachi, and C. Sibertin-Blanc. Protocol Moderators as Active Middle-Agents in Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 8(2) :131–164, 2004. (Cité page 48.)

[16] C. Ming, S. Yuming. Agent Based Intelligent Transportation Management System. In *6th International Conference on ITS Telecommunications Proceedings*, pages 190 – 193. IEEE Computer Society, 2006. (Cité page 87.)

[17] C. Morley, J. Hugues, B. Leblanc, and O. Hugues. *Processus métiers et SI : Evaluation, modélisation et mise en oeuvre*. Dunod, 2002. (Cité pages 2, 17, 18, 19 et 20.)

[18] C. Preist, A. Byde, C. Bartolini, and G. Piccinelli. Towards Agent-based Service Composition through Negotiation in Multiple Auctions. Technical report, Hewlett Packard, 2001. (Cité page 3.)

[19] C. Huang C. Trappey, A. Trappey and C. Ku. The design of a JADE-based autonomous workflow management system for collaborative SoC design. *Expert Syst. Appl.*, 36(2) :2659–2669, 2009. (Cité page 97.)

[20] M. Charfeddine. La logique temporelle. Technical report, UFR Sciences, Université de Nice, 2004. (Cité page 38.)

[21] Y. Charif. *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs*. PhD thesis, Université Pierre et Marie Curie - Paris 6, 2007. (Cité pages 3 et 24.)

[22] Coloane :. http ://move.lip6.fr/software/coloane/. (Cité pages 64 et 94.)

[23] CPN-AMI :. http ://move.lip6.fr/software/cpnami/. (Cité pages 7 et 64.)

[24] D. Benmerzoug, F. Kordon, and M. Boufaida. A Petri-Net based Formalisation of Interaction Protocols applied to Business Process Integration. In *Advances in Enterprise Engineering I, 4th International Workshop on Enterprise & Organizational Modeling and Simulation (EOMAS'08)*, volume 10 of *LNBIP*, pages 78–92, Montpellier, France, June 2008. Springer. (Cité pages 47 et 106.)

[25] D. Benmerzoug, F. Kordon, and M. Boufaida. Formalisation and Verification of Interaction Protocols for Business Process Integration : a Petri net Approach. *International Journal of Simulation and Process Modelling*, 4(3–4) :195–204, 2008. (Cité pages 7, 47 et 106.)

[26] D. Benmerzoug, M. Boufaida, and F. Kordon. A Specification and Validation Approach for Business Process Integration based on Web Services and Agents. In *Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, In conjunction with ICEIS 2007*, pages 163–168. NSTIIC press, 2007. (Cité pages 3, 24 et 105.)

[27] D. Benmerzoug, M. Boufaida, and Z. Boufaida. Developing Cooperative Information Agent-Based Systems with the AMCIS Methodology. In *IEEE International Conference on Advances in Intelligent Systems : Theories and Application*, Luxembourg, November 2004. IEEE press. (Cité pages 87 et 98.)

[28] D. Benmerzoug, S. Djaaboub. Engineering Cooperative JADE Agent with the AMCIS Methodology : The Transportation Management case Study. In *Conférence International sur l'Informatique et ses Applications (CIIA'06)*, page 52, Saida - Algérie, 2006. (Cité page 87.)

[29] D. Benmerzoug, Z. Boufaida, and M. Boufaida. From the Analysis of Cooperation Within Organizational Environments to the Design of Cooperative Information Systems : An Agent-Based Approach. In *OTM Workshops*, volume 3292 of *LNCS*, pages 495–506, Larnaca, Chypre, October 2004. Springer. (Cité pages 29 et 87.)

[30] D. Berthier, C. Morley, and M. M. Demourioux. Enrichissement de la modélisation des processus métiers par le paradigme des systèmes multi agents. Technical report, INT/GET (Groupe des ecoles des Télécommunications), France, 2006. (Cité pages 15 et 45.)

[31] D. Booth, H. Haas, and D. Orchard. Web Services Architecture. Technical report, W3C Group, 2004. 11. (Cité page 75.)

[32] P. Denning. *Work in a Closed-Loop Process*. American Scientist, 1992. (Cité page 46.)

[33] F. Dignum. Social Interactions of Autonomous Agents : Private and Global Views on Communication. In *Formal Models of Agents, ESPRIT Project ModelAge Final Workshop, Selected Papers*, Lecture Notes in Computer Science, pages 103–122. Springer, 1997. (Cité page 39.)

[34] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1. Technical report, World Wide Web Consortium, Mars 2001. (Cité pages 2 et 23.)

[35] E. N. Herness, R.J. High, and J. R. McGee. WebSphere Application Server : A foundation for on demand computing. *IBM Systems Journal*, 43(2) :213–237, 2004. (Cité page 19.)

[36] E. Oliveira, K. Fisher, and O. Stepankova. Multi-Agent Systems : Which Research for which Applications. *Robotics and Autonomous Systems*, 27 :213–261, 1990. (Cité page 29.)

[37] EIF :. *European Interoperability Framework for pan-European eGovernment Services*. Interoperable Delivery of European eGovernment Services to public Administrations, Businesses and Citizens (IDABC), Luxembourg, November 2004. (Cité page 14.)

[38] EU :. The Role of eGovernment for Europe's Future. Technical report, COM(2003), September 2003. (Cité page 14.)

[39] F. Marc, A. El Fallah-Seghrouchni, I. Degirmenciyan. Distributed coordination based on temporal planning for tactical aircraft simulation. http ://perso.club-internet.fr/marc.frederic/aamas03.pdf, 2003. (Cité page 45.)

[40] F. Zambonelli, N.R Jennings, A. Omicini, and M. Wooldridge. Agent-Oriented Software Engineering for Internet Applications. In *Coordination of Internet Agents : Models, Technologies, and Applications*, pages 326–346. 2001. (Cité page 25.)

[41] A. El Fallah-Seghrouchni. *Modèles de coordination d'agents cognitifs*. Principes et architecture des systèmes multi-agents, Hermès, Lavoisier, 2001. (Cité page 73.)

[42] B. Farwer. Modelling protocols by object-based petri nets. In editor L. Czaja, editor, *Proceedings of International conference on Concurrency Specification and Programming (CSP'01)*, pages 87 – 96, 2001. (Cité page 36.)

[43] J. Ferber. *Les Systèmes Multi-Agents : vers une intelligence collective*. Inter-Editions, 1995. (Cité pages 25, 26 et 48.)

[44] FIPA. Foundation for Intelligent Physical Agents : Communicative Act Library Specification, 1997. (Cité pages 28, 76, 80 et 100.)

[45] M. Fisher. A Survey of Concurrent METATEM - the Language and its Applications. In *First International Conference on Temporal Logic*, Lecture Notes in Computer Science, pages 480–505. Springer, 1994. (Cité page 39.)

[46] Foundation for Intelligent Physical Agents. FIPA Contract Net Interaction Protocol Specification. Technical report, FIPA TC Communication, December 2002. (Cité pages x et 32.)

[47] L. Frank. Web services flow language (wsfl 1.0). http ://www-3.ibm.com/software/solutions/, 2001. (Cité page 15.)

[48] G. Alexandre, L. Wilson, S. Siham. Intégration des processus métiers dans les systèmes d'information d'entreprise. Technical report, France Telecom, 2002. (Cité pages 2, 17, 18 et 20.)

[49] H. Kuno G. Alonso, F. Casati and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer-Verlag, 2004. (Cité page 21.)

[50] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modelling Language for Object-Oriented Development, Document Set Version 1.0*. Rational Software Corporation, Santa Clara, 2002. (Cité page 31.)

[51] G. Grossmann, Y. Yang, M. Stumptner, M. Schrefl, and T. Reiter. Analysis of business process integration in Web service context. *Future Generation Comp. Syst.*, 23(3) :283–294, 2007. (Cité page 72.)

[52] G. Mentzas,C. Halaris, and S. Kavadias. Modelling business processes with workflow systems : an evaluation of alternative approaches. *International Journal of Information Management*, 21 :123–135, 2001. (Cité page 45.)

[53] G. Vauvert, and A. El Fallah-seghrouchni. Coalition formation for egoistic agents. In *Int. ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-commerce*, Wollongong, Australia, 2000. (Cité page 72.)

[54] F. Georgel. *IT Gouvernance.* Dunod, 2005. (Cité page 16.)

[55] H. Mazouzi, A. Fallah-Seghrouchni, and S. Haddad. Open Protocol Design for Complex Interactions in Multi-Agent Systems. In *AAMAS'02 : Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 517–526, New York, NY, USA, 2002. ACM. (Cité pages 37, 45 et 47.)

[56] H. Weigand, and W.J van den Heuvel. Cross-organizational workflow integration using contracts. *Decision Support Systems*, 33(3) :247–265, 2002. (Cité page 45.)

[57] S. Haddad and D. Poitrenaud. Modelling and analyzing systems with recursive Petri nets. In *Proc. of the Workshop on Discrete Event Systems - Analysis and Control*, pages 449 – 458, Belgique, 2000. Kluwer Academics. (Cité page 36.)

[58] A. Haddadi. Towards a Pragmatic Theory of Interactions. In *Proceedings of the First International Conference on Multiagent Systems*, pages 133–139, 1995. (Cité page 35.)

[59] M.-P. Huget. *Une ingénierie des protocoles d'interaction pour les systèmes multiagents*. PhD thesis, Université de Paris IX - Dauphine, 2001. (Cité page 4.)

[60] M.N. Huhns. Agents as Web Services. *IEEE Internet Computing*, 6(4) :93–95, 2002. (Cité page 3.)

[61] M.N. Huhns and L.M. Stephens. Multiagent Systems and Societies of Agents. *AAMMAS*, X(X) :79 – 120, 1999. (Cité page 4.)

[62] I. Horrocks, P. Schneider, and M. Dean. SWRL : A semantic web rule language combining OWL and RuleML. Technical report, W3C Submission, May 2004. (Cité page 46.)

[63] I. Paik , S. Takami, and F. Watanabe. Intelligent agent to support design in supply chain based on semantic web services. In *Proceedings of the 4th International Conference on Hybrid Intelligent Systems*,

pages 254–259, Los Alamitos, CA, USA, 2004. IEEE Computer Society. (Cité page 45.)

[64] IBM, Microsoft, SAP, Siebel Systems. Business process execution language for web services version 1.1. Technical report, 2003. (Cité pages 7, 15, 44, 46, 47 et 52.)

[65] ITU-T. Specification and Design Language (SDL). Technical report, , 1996. (Cité page 35.)

[66] J. Barjis, U. Ultes-Nitsche, and J.C. Augusto. Towards more adequate EIS. *Journal of the Science of Computer Programming*, 65(1) :1–3, 2007. (Cité page 13.)

[67] J. Fernandez, C. Jard, T. Jeron, and C. Viho. Using On-The-Fly Verification Techniques for the Generation of test Suites. In *Computer Aided Verification, 8th International Conference CAV'96*, volume 1102 of *Lecture Notes in Computer Science*, pages 348–359, New Brunswick, NJ, USA, 1996. Springer. (Cité page 35.)

[68] T. Jarraya. *Réutilisation des protocoles d'interaction et Démarche orientée modèles pour le développement multi-agents*. PhD thesis, Université de Reims Champagne Ardenne, Décembre 2006. (Cité page 69.)

[69] G. Jean. *L'urbanisation du business et des SI*. Hermès, 2000. (Cité page 16.)

[70] N. R. Jennings. An agent-based approach for building complex software systems. *Commun. ACM*, 44(4) :35–41, 2001. (Cité page 25.)

[71] K. Jensen. *Coloured petri nets : Basic concepts, analysis methods and pratical use*, volume 1. Springer, 1997. (Cité page 94.)

[72] Le journal du net. :. http ://www.journaldunet.com/. (Cité page 20.)

[73] J.P. Courtiat, P. Dembinski, R. Groz, and C. Jard. Estelle :un langage ISO pour les algorithmes distribues et les protocoles. Technical report, INRIA, Rennes - France, 1986. (Cité page 35.)

[74] F. Kordon and P Emmanuel. Using CPN-AMI to validate a safe channel protocol. In *The toolset proceedings of the International Conference on Theory and Applications of Petri Nets*, Williamsburg, Virginia, USA, 1999. (Cité page 94.)

[75] R. Dieter L. Frank. Business Processes in a Web Services World. IBM developerworks, August 2002. (Cité page 15.)

[76] L. Fuhua, N. Douglas, S. Weiming, and R. Kremer. A Schema-Based Approach to Specifying Conversation Policies. In *Issues in Agent Communication*, pages 193–204, London, UK, 2000. Springer-Verlag. (Cité page 37.)

[77] C. Longépé. *Le projet d'urbanisation du système d'information*. 2e édition, Dunod, 2004. (Cité page 16.)

[78] M. Barbuceanu, and M. S. Fox. COOL : A Language for Describing Coordination in Multi Agent Systems. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, pages 17–24. The MIT Press, 1995. (Cité page 28.)

[79] M. Chang, and C. Woo. A Speech-Act-Based Negotiation Protocol : Design, Implementation, and Test Use. *ACM Transactions on Information Systems*, 12(4) :360–382, 1994. (Cité page 46.)

[80] M. Labrousse, and A. Bernard. *Modéliser et gérer les objets d'entreprise : des concepts au système d'information*. GDR MACS, Nantes, 2004. (Cité page 48.)

[81] M. Luck , P. McBurney, O. Shehory, and S. Willmott. The AgentLink Community : Agent Technology : Computing as Interaction A Roadmap for Agent-Based Computing. In *AgentLink III*, 2005. (Cité page 4.)

[82] M. MacKenzie, K. Laskey, F. McCabe, P. Brown, and R. Metz. Reference Model for Service-Oriented Architecture 1.0. Technical report, OASIS. (Cité pages 1 et 2.)

[83] M. Matskin, P. Kungas, J. Rao, J. Sampson, and S. A. Petersen. Enabling Web Services Composition with Softward Agents. In *Proceedings of the Ninth IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA 2005)*, pages 93 – 98, 2005. (Cité page 3.)

[84] M. Munier, K. Baina, and K. Benali. A Negotiation model for CSCW. In Springer Verlag, editor, *LNCS*, volume 1901, pages 224–235, 2000. (Cité page 46.)

[85] M. O Shafiq, Y. Ding, and D. Fensel. Bridging Multi Agent Systems and Web Services : towards interoperability between Software Agents and Semantic Web Services. In *Tenth IEEE International Enterprise Distributed Object Computing Conference*, pages 85–96. IEEE Computer Society, 2006. (Cité page 71.)

[86] M. Singh, and M. Huhns. Automated Workflows for Service Provisioning :Integrating AI and Database Technologies. *IEEE Expert*, 9(5), 1994. (Cité page 46.)

[87] M. Venkatraman and P.S Munindar. Verifying compliance with commitment protocols. *Int. Journal of Autonomous Agents and Multi-Agent Systems*, 2(3) :217 – 236, 1999. (Cité page 50.)

[88] J.P. Meinadier. *Le métier d'intégration des systèmes*. Lavoisier-Hermes, 2002. (Cité page 14.)

[89] T. Melliti. *Interopéerabilité des services Web complexes. Application aux systèemes multi-agents*. PhD thesis, Université Paris IX Dauphine, 2004. (Cité pages 3 et 24.)

[90] P. M. Merlin. *A study of the recoverability of computing systems*. PhD thesis, University of California, Irvine,CA, 1974. (Cité page 36.)

[91] N. Mitra. SOAP Version 1.2 Part 0 Primer. (http ://www.w3.org/TR/2002/CR-soap12-parto-20021219), Décembre 2002. (Cité page 23.)

[92] M.P. Papazoglou, and B. Kratz. Web services technology in support of business transactions. *Int. Journal of Service Oriented Computing*, 1(1) :51 – 63, 2007. (Cité page 24.)

[93] N. Desai, U Ashok, K. Amit, and P. S. Munindar. OWL-P : A Methodology for Business Process Development. In *Agent-Oriented Information Systems III*, volume 3529, pages 79–94. Springer Heidelberg, 2006. (Cité page 46.)

[94] N. Karacapilidis and A. Lazanas and G. Megalokonomos and P. Moraitis. On the development of a web-based system for transportation services. *Inf. Sci.*, 176(13) :1801–1828, 2006. (Cité page 87.)

[95] O. Gutknecht, and J. Ferber. The MADKIT Agent Platform Architecture. In *Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems*, pages 48–55, London, UK, 2001. Springer-Verlag. (Cité page 97.)

[96] P. Bernus, K. Mertins, and G. Schmidt. *Handbook on Architectures of Information Systems*. Springer Verlag, 1998. (Cité page 14.)

[97] P. Bonnet. Cadre de Référence Architecture SOA - Meilleures Pratiques. Technical report, Orchestra Networks, Paris, 2005. (Cité page 21.)

[98] P. Populaire, Y. Demazeau, O. Boissier, and J. Sichman. Description et implémentation de protocoles de communication en univers multi-agents. In *1ères journées Francophones sur l'Intelligence Artificielle et les Systèmes Multi-Agents*, Toulouse, 1993. (Cité page 35.)

[99] P.A. Buhler, and J.M. Vidal. Towards adaptive workflow enactment using multiagent systems. *International Journal On Information Technology and Management*, 6 :61–87, 2005. (Cité page 45.)

[100] M. P. Papazoglou. The Role of Agent Technology in Business to Business Electronic Commerce. In *Cooperative Information Agents III, Third International Workshop*, volume 1652 of *Lecture Notes in Computer Science*, pages 245–264. Springer, 1999. (Cité page 71.)

[101] M. P. Papazoglou. Agent-oriented technology in support of e-business. *Commun. ACM*, 44(4) :71–77, 2001. (Cité page 73.)

[102] H. V. Parunak. Visualizing Agent Conversations : Using Enhanced Dooley Graphs for Agent Design and Analysis, 1996. (Cité pages x, 30 et 31.)

[103] C. Petri. Fundamentals of a theory of asynchronous information flow. In *Proceeding of the 1962 IFIP Congress*, pages 386–390, Amsterdam, North-Holland, 1962. (Cité page 36.)

[104] B. Philippe. *XTL : une logique temporelle pour la spécification formelle des systèmes interactifs*. PhD thesis, Université Paris XI Orsay, 1998. (Cité page 38.)

[105] S. Dustdar P.M. Papazoglou, P. Traverso and F. Leymann. Service-oriented computing : a research roadmap. *nt. Journal of Cooperative Information Systems*, 17(2) :223 – 255, 2008. (Cité pages 3 et 24.)

[106] R. C. Philip, and H. J. Levesque. Intention is Choice with Commitment. *Artificial Intelligence*, 42(2–3) :213–261, 1990. (Cité page 27.)

[107] R. Hamid, N. Motahari, B. Benatallah, F. Casati, and F. Toumani. Web Services Interoperability Specifications. *IEEE Computer*, 39(5) :24–32, 2006. (Cité pages x, 21 et 22.)

[108] R. Kishore, H. Zhang, and R. Ramesh. Enterprise integration using the agent paradigm : foundations of multi-agent-based integrative business information systems. *Decison Support Systems*, 42(1) :48–78, 2006. (Cité pages 45 et 47.)

[109] R. Medine-Mora, T. Winograd, and F. Flores. The Action Workflow Approach to Workflow Management Technology. In *Int. Conf. On Computer Supported Cooperative Work*, 1992. (Cité page 46.)

[110] R. N. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1) :7–38, 1998. (Cité page 25.)

[111] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng. Modeling Agent Conversations with Colored Petri Nets, 1999. (Cité page 37.)

[112] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974. (Cité page 36.)

[113] R. Reix. *Systèmes d'information et management des organisations*. Vuibert, 2002. (Cité page 16.)

[114] G. Ribière. Communication et traitement en mode message avec MQSeries. Technical Report H2-768, Techniques de l'ingénieur, 1998. (Cité page 19.)

[115] S. Aknine, S. Pinson, and M. Shakun. An Extended Multi-Agent Negotiation Protocol. *Int. Journal on Autonomous Agents and Multi-agent Systems*, 8(1) :1–45, 2004. (Cité page 46.)

[116] S. Vinoski. Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communication magazine*, 35(2), February 1997. (Cité page 18.)

[117] T. Satich. XLANG - Web Services For Business Process Design. http ://www.gotdotnet.com/xlang-c/default.htm, 2001. (Cité page 15.)

[118] J. Searle. *Speech Acts : an Essay in the Philosophy of Language*. Cambridge Press, 1969. (Cité page 27.)

[119] Y. Seo and K. Sycara. Exploiting Multi-Agent Interactions for Identifying the Best-Payoff Information Source. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 344–350. IEEE Computer Society, 2005. (Cité pages 4 et 28.)

[120] D. Serain. *Enterprise application integration*. Dunod, Paris, 3ème edition, 2001. (Cité page 1.)

[121] R. G. Smith. The Contract Net Protocol : High Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computing*, 29(12) :1104–1113, 1980. (Cité page 46.)

[122] J. M. Spivey. *The Z notation : a reference manual*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. (Cité page 35.)

[123] SWARMS :. (http ://www.swarm.org/index.html), 1996. (Cité page 97.)

[124] T. Bellwood, L. Clément, and C. von Riegen. Universal Description, Discovery and Integration. Technical report, OASIS UDDI Specification Technical Committee, Mars 2002. (Cité pages 2 et 23.)

[125] T. Bolognesi, and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1) :25–29, 1988. (Cité page 35.)

[126] T. W. Finin, R. Fritzson and D. P. McKay, and R. McEntire. KQML As An Agent Communication Language. In *CIKM*, pages 456–463. ACM, 1994. (Cité page 28.)

[127] T. W. Finin, R. Fritzson, D. P. McKay, and R. McEntire. KQML As An Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463. ACM, 1994. (Cité pages 80 et 100.)

[128] C. Tanguy. Business Process Management : De la modélisation à l'exécution, Positionnement par rapport aux Architectures Orientées Services. Livre blanc, Intalio, 2003. (Cité pages 14 et 15.)

[129] O. V. Tschammer. *An Agent-Based Platform for the Management of Dynamic Virtual Enterprises*. PhD thesis, Berlin, 2001. (Cité page 73.)

[130] V. Ermolayev, M. Keberle, and S. Plaksin. Towards Agent-Based Rational Service Composition - RACING Approach. In *ICWS-Europe*, volume 2853 of *LNCS*, pages 167–182. Springer, 2003. (Cité page 72.)

[131] F.B. Vernadat. *Enterprise Modelling and Integration : Principles and Applications*. 1996. (Cité pages 13 et 14.)

[132] W. Jessica, J. Barjis, A. Verbraeck, M. Janssen, and J. Kort. Capturing Complex Business Processes Interdependencies Using Modeling and Simulation in a Multi-actor Environment. In *CIAO ! / EOMAS*, volume 34 of *Lecture Notes in Business Information Processing*, pages 16–27. Springer, 2009. (Cité page 48.)

[133] W. M. P. van der Aalst and A. H. M. Hofstede and W. Mathias. Business Process Management : A Survey. In *International Conference on Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003. (Cité page 14.)

[134] G. Wagner. The Agent-Object-Relationship Metamodel : towards a unified view of state and behavior. *Information Systems*, 28 :475–504, 2003. (Cité page 45.)

[135] R. H. Weston. Steps Towards Enterprise Wide Integration : A Definition of Needs and First Generation Open Solutions. *International Journal of Production Research*, 31(5) :2235–2254, 1993. (Cité page 13.)

[136] T.J. Williams. The Purdue Enterprise Reference Architecture. *Computers in Industry*, 24(2-3), 1994. (Cité page 14.)

[137] M. Wooldridge and M. Fisher. A Decision Procedure for a Temporal Belief Logic. In *First International Conference on Temporal Logic*, Lecture Notes in Computer Science, pages 317–331. Springer, 1994. (Cité page 39.)

[138] Y. Hoffner, H. Lwdwig, C. Gulcu, and P. Grefen. An architecture for crossorganisational business processes. In *Advanced Issues of E-Commerce and Web-Based Information Systems. WECWIS*, pages 62 – 72. IEEE computer society, 2000. (Cité page 45.)

[139] Y. Labrou, and T. W. Finin. A Semantics Approach for KQML - A General Purpose Communication Language for Software Agents. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 447–455. ACM, 1994. (Cité page 28.)

[140] Y. Labrou, T. Finin. A Proposal for a New KQML Specification. Technical report, Université de Maryland Baltimore, 1997. (Cité page 28.)

[141] N. Zarour. *Contribution à la modélisation de la coopération des systèmes d'information distribués et hétérogènes : le système DAArACHE - Application aux entreprises de production*. PhD thesis, Université de Constantine, Septembre 2004. (Cité pages 1 et 72.)

Annexes

A

Sommaire

A.1 L'outil IP2CPN

Dans cette section, nous présentons un outil automatique pour la traduction des PI en RdPC. l'outil IP2CPN permet de générer un fichier PetriScript à partir d'une spécification d'un PI comme elle a été défini dans la section 3.2 du chapitre 2 (voir fig. A.1).

FIGURE A.1 – *L'outil IP2CPN*

La génération de RdPC se fait à partir d'une spécification PI en respectant les règles de passages que nous les introduisons dans le chapitre 2. Le RdPC est construit en plusieurs itérations : l'algorithme permet de générer le reseau de Petri en explorant le protocole d'interaction.

la figure A.2 montre le diagramme de classe de l'outil IP2CPN. Un protocole d'interaction est un ensembles non vide des partenaires (les intervenants dans l'interaction) et un ensemble non vide des messages échangés durant cette interaction.

Un message peut etre un message simple (ou primitif) ou un message complexe. Il est à noter qu'un message complexe est composé à partir des messages simples par le moyen des opérateurs OR, AND ou XOR.

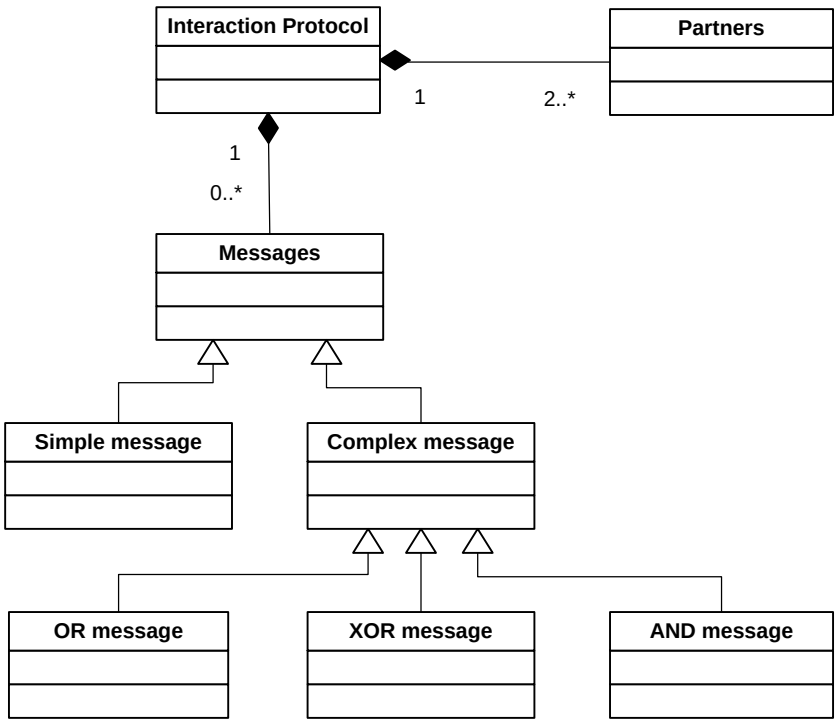

FIGURE A.2 – *Diagramme de classe de l'outil IP2CPN*

Pour simplfier l'implementation de l'oultil IP2CPN, nous introduisons le concept de *la matrice de progression*. *la matrice de progression* est un tableau dynamique et a deux dimension.

```
ArrayList<ArrayList<String> > ProgressMatrix= new
ArrayList<ArrayList<String> >();
```

Cette structure de donnée est initialisée par les places qui correspondent aux partners du protocole. Chaque fois l'outil IP2CPN traite un message, *la matrice de progression* est mise-à-jours.

La figure A.3 montre la génération de RdPC (sous format PetriScript) ainsi que *la matrice de progression* correspondante à cette protocoles d'interaction.

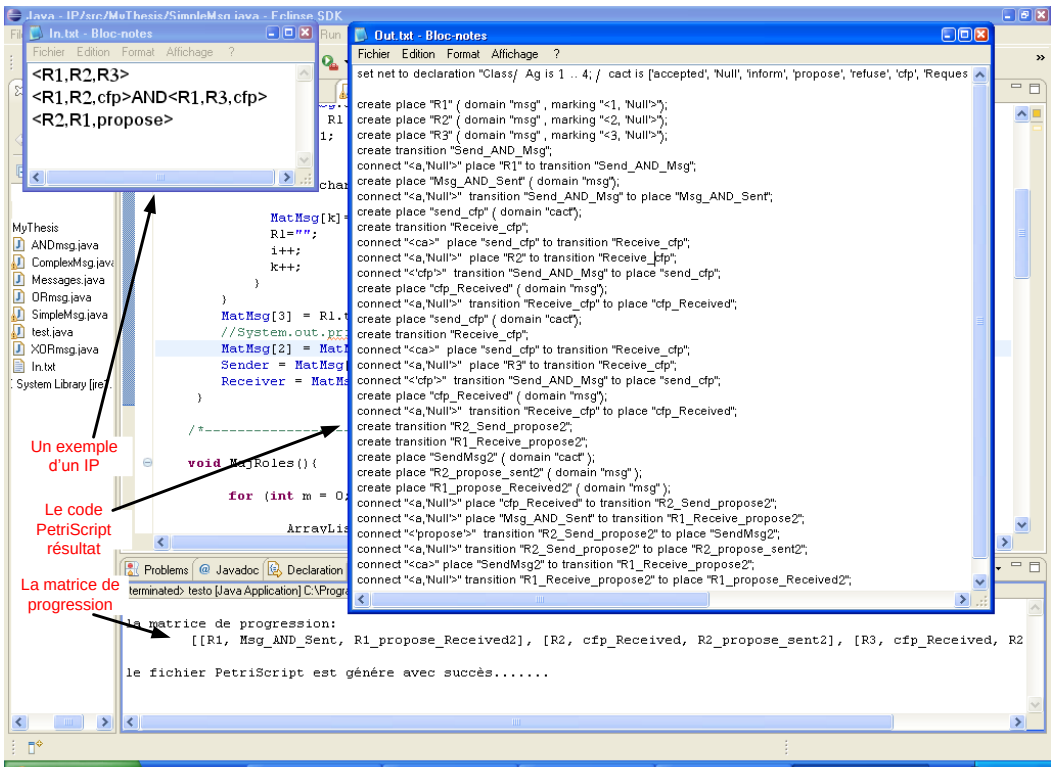

FIGURE A.3 – *Exemple de génération de code avec IP2CPN*

concepts d'architecture	code JADE
Agent	Jade.core.agent Class
Agent plan	FSMBehaviour Class
Agent capability	OneShotBehaviour Class
Perception interface	CyclicBehaviour Class
Plan Retrieval	SimpleBehaviour Class
Message	ACL Message Class

TABLE A.1 – *Règles d'implémentation de l'architecture avec la plate-forme JADE*

A.2 Implémentation des agents dans la plate-forme JADE

A.2.1 Correspondance entre les concepts de l'architecture et Jade

Comme nous avons présenté dans le chapitre 5 et afin de réaliser une opérationnalisation plus systémique de notre architecture, le développeur se base sur un ensemble des règles permettant d'implémenter facilement les concepts clés (plans, capacités, protocoles d'interaction, etc.) de notre architecture. Nous rappelons que ces règles (tableau A.1 s'appliquent seulement dans l'environnement JADE.

Dans le reste de cette annexe, nous essayerons de présenter le résultat d'implémentation de notre architecture a l'aide des interfaces graphiques (GUI : Graphical User Interface) proposer par la plateforme JADE.

A.2.2 Exécution de l'application à l'aide de la plateforme JADE

Pour exécuter l'application, nous avons besoin de l'environnement JADE version 1.3, ainsi que l'outil JDK 1.5. L'environnement auquel accède l'utilisateur est illustré par la figure A.4. Cette figure correspond à l'interface graphique (GUI) de l'agent RMA (Remote Monitoring Agent) dont son rôle est de contrôler et superviser la plate-forme ; il est responsable de l'authentification et l'enregistrements des agents du système.

L'agent RMA a plusieurs fonctionnalités, en effet, il nous permet de

- Lancer un nouvel agent : l'utilisateur peut spécifié le conteneur sur lequel l'agent est lancé. Si aucun conteneur n'est spécifié, l'agent est lancé sur le conteneur principal de la plate-forme. Comme l'indique la figure A.5, l'utilisateur saisit le non de l'agent (ServiceClientele dans notre exemple), ainsi que le chemin de la classe Java de cet agent (la classe qui hérite la classe de base Jade.core.Agent).
- Suspendez l'exécution d'un agent choisi : cette action suspende l'exécution de l'agent choisi. Elle est équivalente à l'appel de la méthode doSuspend().
- Résumer l'exécution d'un agent suspendu : cette action met les agents suspendu en un état actif.

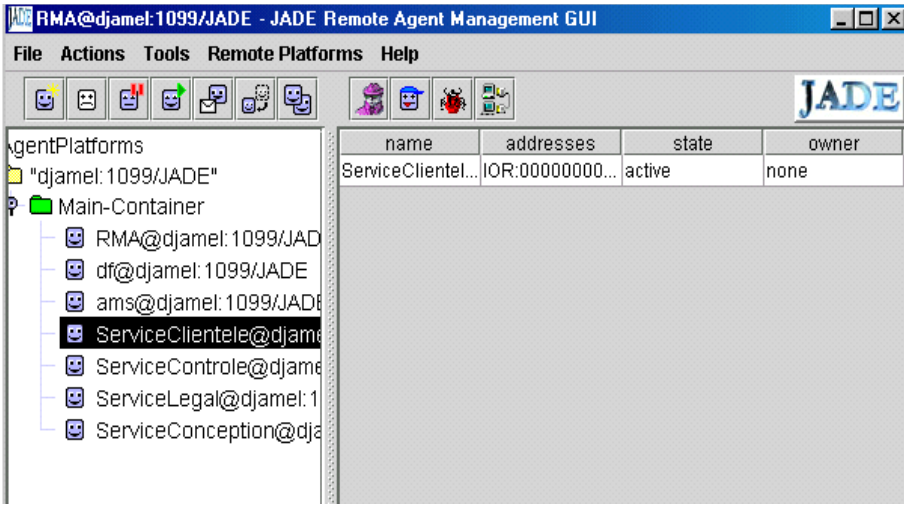

FIGURE A.4 – *Le GUI de l'agent RMA*

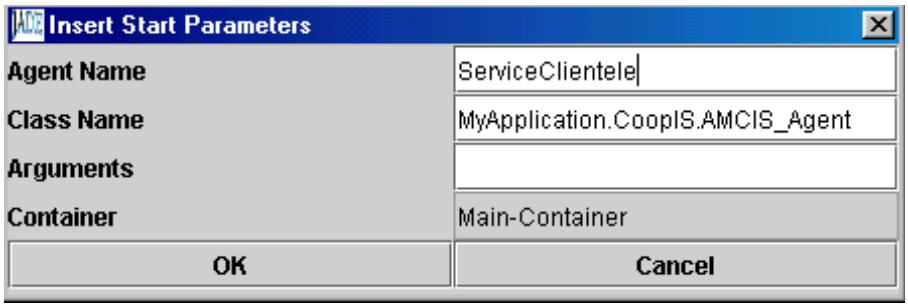

FIGURE A.5 – *Boite de dialogue montrant l'ajout d'un nouvel agent*

– Envoyez un message à un agent Choisi : cette action permet d'envoyer un message ACL à un agent choisi. Quand l'utilisateur choisit cet agent de menu, une boite de dialogue est montrée dans lequel un message ACL peut être composé et envoyé. La figure A.6-(a) montre la demande du prix d'un plan envoyée par l'agent ServiceClientele à l'agent ServiceConception et la figure A.6-(b) montre la réponse à cette demande.

A.2.3 La gestion de la liste des messages ACL avec l'agent Dummy

Cet agent offre un GUI qui permet de composer et envoyer des messages ACL. En plus il maintient une liste de tous les messages ACL envoyés et reçus. Cette liste peut être examinée par l'utilisateur. En outre, la liste de message peut être sauvegardée et récupérée plus tard. La figure A.7 montre l'interface de l'agent Dummy.

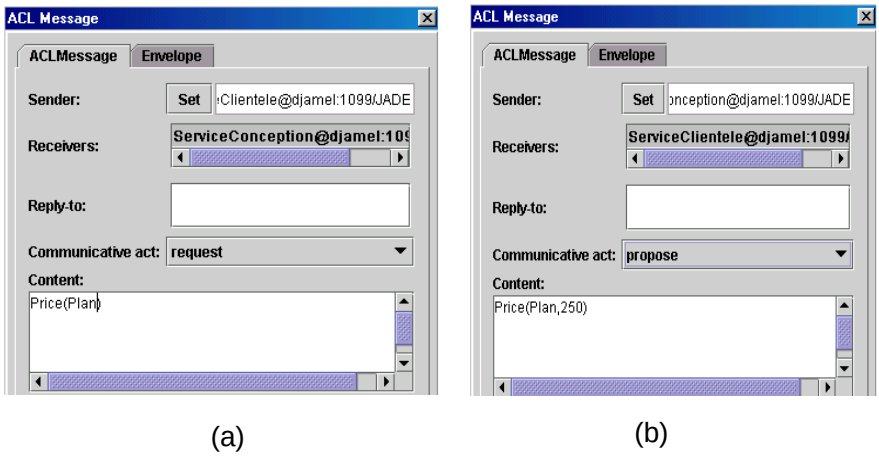

FIGURE A.6 – *Boite de dialogue montrant la représentation d'un message ACL*

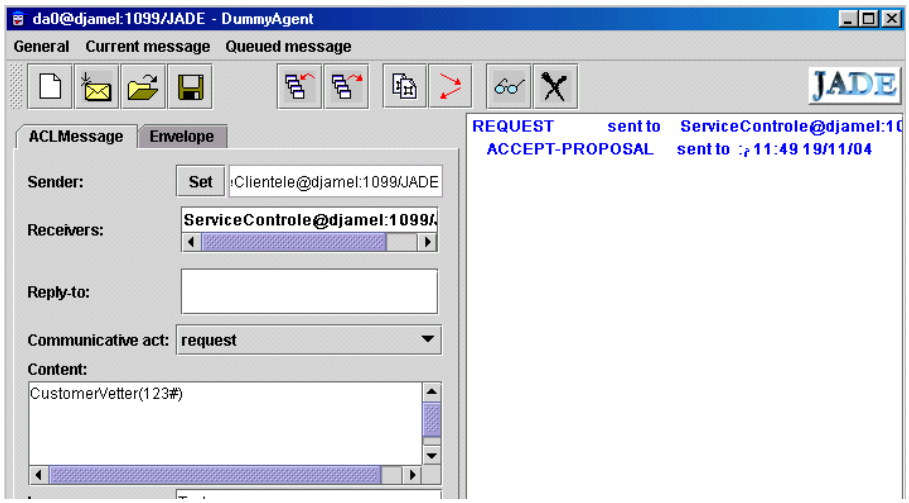

FIGURE A.7 – *Le GUI de l'agent Dummy*

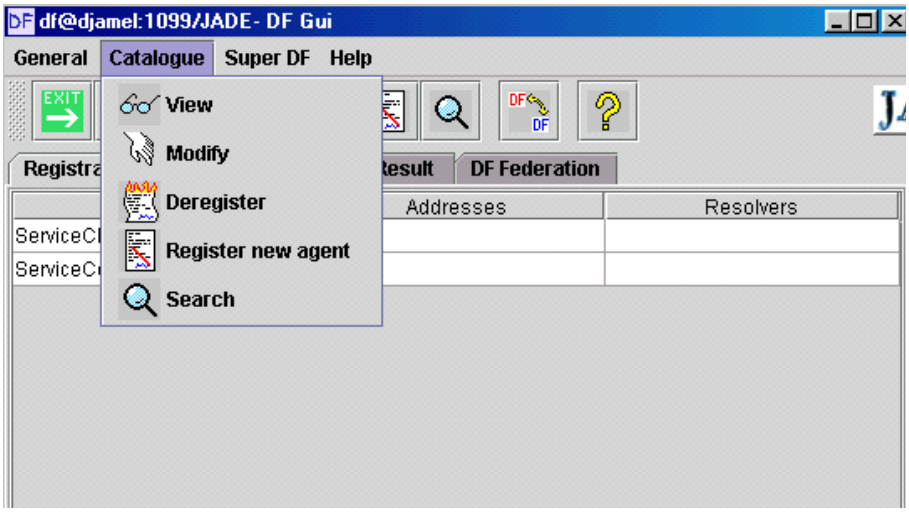

FIGURE A.8 – *Le GUI de l'agent DF*

A.2.4 Le contrôle des descriptions d'agents avec l'agent DF

DF (Facilitateur d'Annuaire) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents. En employant ce GUI (figure A.8), l'utilisateur peut agir réciproquement avec le DF : considérer les descriptions des agents enregistrés, enregistrer ou supprimer des agents, modifier le descriptions des agents enregistrés et chercher aussi des descriptions d'agent.

A.2.5 La visualisation des messages échangés avec l'agent Sniffer

Cet agent offre la possibilité de suivre la communication entre les agents de système. Dans ce cas, cet agent peut visualiser et sauvegarder les messages échangés. La figure A.9 montre les messages échangés entre les agents de notre application.

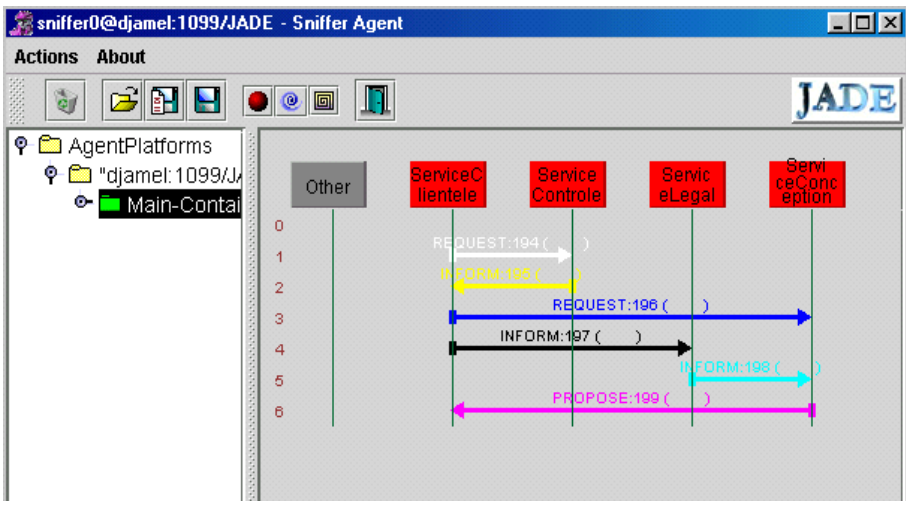

FIGURE A.9 – *Le GUI de l'agent Sniffer*

نماذج و أدوات شكلية لإدماج تطبيقات الشركات

إدماج تطبيقات الشركات تتعلق أساسا بالتفاعلات الموجودة بين مختلف الإجراءات اللازمة لانجاز هدف مشترك. و عليه فان التفاعلات هي عامل رئيسي ومهم من أجل تحفيز الاشتراك في المنابع و تنظيم مهام مختلف تطبيقات الشركات. في هاته الرسالة نقترح استعمال مفهوم بروتكولات التفاعلات من اجل تمثيل وتصميم إدماج تطبيقات الشركات. وعليه فقد استعملنا اللغة التمثيلية AUML واللغة النصية BPEL4WS . أيضا الطريقة المقترحة في هاته الرسالة تعتمد على شبكات Petri من أجل التحقق من صحة البروتكولات التفاعلية. فيما يخص البرمجة فقد استعملنا عدة لغات من أجل إنشاء النظام المقترح.

FIGURE A.10

Titre Modèles et outils formels pour l'intégration d'applications d'entreprises

Résumé L'intégration d'applications dépeint les interactions dans lesquelles les processus engagés participent afin d'atteindre un objectif commun, ainsi que les dépendances entre les interactions. En effet, l'interaction est un mécanisme important pour supporter le partage des ressources et coordonner les activités entre les différentes applications de l'entreprise. Le concept de protocole d'interaction est un moyen efficace pour structurer et organiser les échanges entre ces applications. Dans ce travail de recherche, nous proposons un cadre conceptuel et une démarche méthodologique couvrant les principales phases pour la modélisation de l'intégration d'applications basée sur les protocoles d'interaction. Cette démarche est basée conjointement sur l'utilisation de la notation AUML, et aussi sur le langage BPEL4WS. Nous définissons par la suite un guide permettant de passer à partir de descriptions semi-formelles des protocoles d'interaction, vers des spécifications formelles de réseaux de Petri colorés (RdPC) tout en mettant l'accent sur le contrôle et la dynamicité des interactions. Le modèle de RdPC permet de couvrir les aspects de modélisation et de validation des protocoles.

Mots-clés Intégration d'applications, Processus métier, Protocole d'interaction, Réseaux de Petri, Vérification et validation

Title Formal models and tools for enterprise application integration

Abstract Interaction Protocols (IP) are specific, often standard, constraints on the behaviours of the autonomous agents in a multiagent system. Protocols are essential to the functioning of open business systems, such as those that arise in most interesting web applications. This thesis presents a new approach for Business Process Integration (BPI) based on IP. It enables both integration and collaboration of autonomous and distributed business processes modules. We present a semantic formalisation of the IP notations used in our approach. The semantics and its application are described on the basis of translation rules to Coloured Petri Nets (CPN) and the benefits of formalisation are shown.

Keywords Business Process Integration, Interaction Protocol, Petri nets, Verification and Validation