**UNIVERSITÉ D'ÉVRY VAL D'ESSONNE**

**UFR SCIENCES ET TECHNOLOGIES**

**IBISC : Laboratoire d'Informatique, Biologie Intégrative et Systèmes Complexes**

**Thèse**

**présentée par**

**Nader CHEAIB**

**pour obtenir**

**LE GRADE DE DOCTEUR EN SCIENCES DE L'UNIVERSITÉ D'ÉVRY VAL D'ESSONNE**

**Spécialité Informatique**

# Contribution à la malléabilité des collecticiels : une approche basée sur les services web et les agents logiciels

Soutenue publiquement le 16 Juin 2010

JURY

| | | |
|---|---|---|
| B. David : | Professeur, Ecole Centrale de Lyon | Rapporteur |
| P. Ghodous : | Professeur, Université Claude Bernard Lyon 1 | Rapporteur |
| P. Le Parc : | Professeur, Université de Bretagne Occidentale | Examinateur |
| A. Dinis : | Coordinateur Projet DigitalOcean, VirtualDive | Invité |
| S. Otmane : | Maître de Conférences, Université d'Evry | Encadrant |
| M. Mallem : | Professeur, Université d'Evry | Directeur de Thèse |

# Remerciements

*Ce travail de recherche a été réalisé au Laboratoire IBISC (Informatique, Biologie Intégrative et Systèmes Complexes), au sein de l'équipe RATC (Réalité Augmentée et Travail Collaboratif), à l'Université d'Evry val d'Essonne, sous la direction du Monsieur MALIK MALLEM, Professeur à l'Université d'Evry Val d'Essonne. Je tiens à le remercier pour m'avoir accueilli dans son équipe et pour m'avoir guidé tout au long de ma thèse. Je remercie l'Agence Nationale de la Recherche (ANR) pour avoir financé ce travail pendant trois ans.*

*Je remercie Monsieur SAMIR OTMANE, Maître de Conférences à l'Université d'Evry Val d'Essonne et responsable scientifique pour IBISC du projet DigitaOcean, pour son encadrement et son soutien tout au long de cette thèse.*

*Je remercie Madame PARISA GHODOUS, Professeur à l'Université Claude Bernand Lyon 1, Monsieur BERTRAND DAVID, Professeur à l'École Centrale de Lyon, Monsieur PHILIPPE LE PARC, Professeur à l'Université de Bretagne Occidentale, et ALAIN DINIS, fondateur de la société VirtualDive et initiateur du projet DigitalOcean, pour avoir accepté de faire partie du jury.*

*Je remercie NICOLAS FIÉS, ingénieur chez VirtualDive, avec qui j'ai collaboré dans le contexte du projet DigitalOcean. Je remercie mes collègues avec qui j'ai passé des moments agréables : MAHMOUD HAYDAR pour nos pauses café, MOUNA ESSABBAH et IMANE ZENDJEBIL pour nos débats passionnants dans le train quotidien, PIERRE BOUDOIN et CHRISTOPHE DOMINGUES pour leurs disponibilités et les moments pleins d'humour. Je remercie CHRISTOPHE MONTAGNE et JEAN-YVES DIDIER avec qui j'ai effectué mes enseignements, ainsi que pour nos discussions très sympathiques. Je remercie enfin tous les membres de l'équipre RATC pour leur acceuil chaleureux.*

*Je remercie HELA SEKMA pour sa présence précieuse durant les derniers mois de la thèse.*

*Finalement, je dédie ce travail à ma famille. Je ne serai pas ici, en train d'écrire ces mots, sans leur soutien et leur amour inconditionnel. Je remercie mon père ALI qui m'a appris à rêver, ma mère LEILA qui m'a appris à être fort dans les moments difficiles, ma sœur DIMA qui m'a appris à surmonter nos douleurs et voir les bons côtés des choses, et ma sœur ROUBA qui m'a appris à connaitre le nouveau monde loin des parents et à compter sur soi. Je vous aime tant.*

# Table des matières

# Table des figures

# Liste des tableaux

# Résumé

L'objectif du TCAO (Travail Collaboratif Assisté par Ordinateur), est de trouver les moyens par lesquels les applications collaboratives sont susceptibles d'améliorer la collaboration entre les individus. De ce fait, il existe une grande nécessité de remédier des contraintes liées à la manque de flexibilité et la rigidité des systèmes collaboratifs actuels, par l'adoption des solutions adéquates pour mettre en œuvre une meilleure collaboration, selon le contexte et la tâche à effectuer entre les utilisateurs. En effet, le domaine du TCAO doit évoluer avec l'évolution des systèmes et des technologies qui touchent notre vie quotidienne, surtout l'évolution de l'internet qui nous rend totalement dépendant des services et applications qui existent "virtuellement", où la plupart des utilisateurs passent une bonne partie de leurs temps à exploiter des méthodes à rechercher et utiliser ces services qui correspondent le plus à leurs préférences. C'est pour cette raison que l'évolution du TCAO se montre essentielle pour faire face à l'évolution exponentielle des technologies d'internet, afin de créer ou de réutiliser plus facilement des applications chargées d'assister le travail communautaire des hommes, que l'on nomme applications collaboratives, ou collecticiels.

Le sujet de thèse proposé couvre les aspects collaboratifs d'un système et les questions concernant son intégration. Plus particulièrement, notre objectif essentiel est de concevoir une architecture logicielle pour les collecticiels malléables, de sorte qu'elle puisse s'adapter aux changements et aux diversités des besoins des utilisateurs, ainsi que la tâche à effectuer. En conséquence, une forte exigence surgit en terme d'ouverture, où le système peut dynamiquement intégrer de nouveaux services sans arrêter le déroulement de la collaboration, ni manuellement recoder et recompiler l'application. Une deuxième exigence est d'assurer une certaine adaptabilité, où le système pourra générer de nouveaux comportements à partir de la composition de deux ou plusieurs services. Finalement, une exigence surgit en terme d'interopérabilité, surtout dans le cas où les utilisateurs utilisent des applications incompatibles ou hétérogènes. Ainsi, la création, l'ajout, la suppression ou la manipulation des composants du système collaboratif seront faites via les services web. De plus, la recherche, l'invocation et l'intégration de ces services se fera à l'aide d'agents logiciels qui se chargeront, avec une assistance minimale de l'utilisateur, de rechercher les services les mieux adaptés à leurs spécifications. Dans cette thèse, nous créons un lien entre les concepts théoriques qui se développent au sein des laboratoires de recherche, et les technologies qui se développent d'une façon très rapide dans le secteur industriel, afin de concevoir des systèmes collaboratifs plus adaptés au monde informatique quotidien.

`Mots Clés` : TCAO (Travail Coopératif Assisté par Ordinateur), services web, agents logiciels, architecture collaborative, interopérabilité, malléabilité des systèmes.

# Abstract

The aim of CSCW (Computer Supported Cooperative Work) is to find ways in which applications should improve collaborative work between individuals. Hence, there is great need to address constraints related to the lack of flexibility and rigidity of current collaborative systems, through the adoption of adequate solutions to implement a better collaboration, depending on user' needs and the task that is being done. Therefore, the field of CSCW must evolve with the evolution of systems and technologies that affects our daily lives, especially the internet evolution that makes us completely dependent on the services and applications that "virtually" exist, where most people spend a lot of their time collaborating and exploiting methods to find and use services that meet their preferences. The development of CSCW systems appears essential to address the exponential growth of internet technologies to create or reuse applications to assist the community work of men, known as collaborative applications, or groupware.

In this work, the thesis covers collaborative aspects of a system, and the questions concerning its integration. More specifically, the main objective is to provide a platform for "tailorable" collaboration, where the services offered by the groupware can be adapted to the changing and diverse needs of users. Accordingly, strong requirements arise in terms of adaptability, by composing or integrating new services without stopping the collaboration process and interoperability between the system's components, especially if users are using incompatible or heterogeneous applications. A proposed solution is to use the concepts of web services, integrated with the concepts of multi-agent systems (MAS). Thus, the creation, addition, deletion or dynamic manipulation of the system's components will be done via the web services. In addition, research, invocation and integration of these services will be done using software agents with minimal user assistance, depending on users' preferences. In this thesis, we try to build a bridge between theoretical concepts which are developed in research laboratories, and technologies being developed exponentially in the industrial sector, hence, creating a synergy of theory and concepts, to design more efficient collaborative systems, that are better suited to the everyday computing world.

`Keywords` : CSCW (Computer Supported Cooperative Work), web services, software agents, groupware architecture, tailorability, interoperability.

# Glossaire

- SOAP : Simple Object Access Protocol
- WSDL : Web Service Description Language
- UDDI : Universal Description Dicovery and Integration
- JADE : Java Agent Development Framework
- DF : Directory Facilitator dans JADE
- WSIG : Web Service Integration Gateway qui est un plugin de la platforme JADE
- TCAO : Travail Collaboratif Assisté par Ordinateur
- U3D : Universal Directory for Description and Discovery
- UDDI4C : Universal Description Dicovery and Integration for Collaboration

# INTRODUCTION GENERALE

## Problématique

L'objectif du TCAO (Travail Collaboratif Assisté par Ordinateur) est de trouver les moyens par lesquels les applications collaboratives sont susceptibles d'améliorer la collaboration entre les individus, tout en mesurant leur impact sur le comportement des groupes. Ainsi, le "travail collaboratif" est effectué en utilisant des artéfacts qui ne cessent d'évoluer en termes de rapidité et de performance, en influençant extrêmement le travail des individus. La distinction essentielle entre le travail collaboratif et le travail mono-utilisateur, est que dans le premier, plusieurs individus travaillent d'une façon synchrone ou asynchrone pour effectuer une tâche commune, en utilisant des applications qui sont souvent hétérogènes. En effet, le domaine du TCAO doit évoluer avec l'évolution des systèmes et des technologies qui touchent notre vie quotidienne. En même temps, l'évolution de l'Internet nous rend totalement dépendant des services qui existent "virtuellement", où la plupart des utilisateurs passent une bonne partie de leurs temps à les exploiter.

C'est pour cette raison que l'évolution du TCAO se montre essentielle pour faire face à l'évolution exponentielle des technologies d'Internet, afin de créer ou de réutiliser plus facilement des applications chargées d'assister le travail communautaire des hommes. Ces types d'applications sont appelées applications collaboratives, ou collecticiels. Dans ce travail, le sujet de thèse proposé couvre les aspects collaboratifs d'un système et les questions concernant son intégration. Plus particulièrement, notre objectif essentiel est d'offrir une plateforme de collaboration "malléable", de sorte qu'elle puisse s'adapter aux changements et aux diversités des besoins des utilisateurs. En conséquence, de fortes exigences surgissent en terme :

- D'ouverture, où les utilisateurs pourront dynamiquement intégrer de nouveaux services/composants, et cela sans arrêter le déroulement de la collaboration, ni recoder manuellement ou recompiler l'application.
- D'adaptabilité, qui est la propriété de générer de nouveaux comportements à partir de composants de bases qui constituent le système.
- D'interopérabilité, surtout dans le cas où les usagées utilisent des applications incompatibles ou hétérogènes.

De ce fait, il est nécessaire d'une part, de prendre en compte ces contraintes lors des phases de modélisation et de conception des architectures logicielles des collecticiels. D'autre part, il est également nécessaire de créer un lien entre les concepts théoriques qui se développent au sein des laboratoires de recherche, et les technologies qui se développent d'une façon très rapide dans le secteur industriel. Ce lien permettrait donc de concevoir des

systèmes collaboratifs plus adaptés au monde informatique quotidien. Ainsi, les collecticiels malléables résultants, contribueraient à améliorer la collaboration entre les individus en leur offrant la possibilité d'adapter les outils collecticiels aux besoins des utilisateurs et des tâches à réaliser.

Dans notre recherche, nous avons constaté que les questions concernant la malléabilité sont plus nombreuses que les réponses, sans un support suffisant pour l'orientation, la comparaison et la classification. Ainsi, fournir une vue générale et compréhensive de ces approches est un support important pour le domaine du TCAO. Nous abordons des questions telles :

- Quelles sont les aspects qui contribuent à la malléabilité des systèmes collaboratifs ?
- Quel modèle d'architecture logicielle pour la modélisation et la conception des collecticiels malléables ?
- Quelles sont les technologies et les outils logiciels facilitant l'intégration et l'implémentation des propriétés de la malléabilité dans les collecticiels ?
- Quel type de malléabilité faut-il pour une performance raisonnable du système collaboratif ?
- Peut-on mesurer la malléabilité d'un collecticiel ? Et comment ?

Dans cette thèse nous tentons de répondre à toutes ces questions avec comme objectif principale la conception d'une architecture logicielle générique comme support de la malléabilité dans les collecticiels.

## Solution Proposée

La solution que nous proposons repose sur une approche d'intégration des concepts des services web et des agents logiciels, afin de concevoir et d'implémenter une architecture logicielle comme support de la malléabilité des collecticiels. Ainsi, nous définissons un collecticiel malléable, comme un logiciel collaboratif qui possède une capacité de créer de nouveaux comportements, par un mécanisme d'orchestration de services. Ce mécanisme offre deux processus principaux pour soutenir la malléabilité dans les collecticiels :

- L'intégration de services web dans le système qui n'étaient pas prévues lors de la phase de conception. Les services ainsi invoqués répondent le plus aux spécifications des utilisateurs.
- La composition de deux ou de plusieurs services, afin de générer un nouveau service avec un nouveau comportement.

Notre travail se base essentiellement sur la collaboration machine-machine. Ainsi, nous proposons de mettre en valeur un cadre de collaborations qui se passent entre deux ou plusieurs machines sur internet, afin d'échanger des services dans un contexte collaboratif. Ainsi, nous mettons la "machine" au centre de la collaboration. Le but est donc d'améliorer la collaboration entre les utilisateurs via cette collaboration machine-machine. Un des avantages de cette approche, est que l'utilisateur ne crée pas le composant logiciel lui-même pour avoir une fonctionnalité demandée. Ainsi, une demande d'une fonctionnalité non prévue lors de la phase de conception, déclenche un de ces deux mécanismes pour la générer. La collaboration machine-machine, par le biais des services web/agents

logiciels, se charge de mettre en œuvre ces mécanismes, avec une assistance minimale des utilisateurs.

## Organisation du manuscrit

Le manuscrit est organisé en deux parties et cinq chapitres. La première partie présente l'état de l'art dans et elle est organisée en deux chapitres. La deuxième partie présente notre contribution pour répondre aux problématiques posées et elle est organisée en trois chapitres.

Le premier chapitre présente un état de l'art sur les modèles d'architecture et des systèmes collaboratifs. Nous commençons par définir le terme collecticiel à partir du modèle de trèfle fonctionnel des collecticiels (Communication, Coordination et Production), qui constitue la base de nos travaux pour concevoir la collaboration logicielle. Ensuite, nous présentons une étude des modèles d'architectures logicielles pour les collecticiels. Puis, nous présentons quelques systèmes collaboratifs qui existent dans la littérature. Finalement, nous classons les types d'architectures présentées selon des propriétés d'adaptabilité, d'ouverture et d'interopérabilité, qui se traduisent par la propriété de la malléabilité.

Le deuxième chapitre présente les technologies et les approches pour l'interopérabilité et la malléabilité des systèmes collaboratifs. Il est divisé en trois sections. Dans la première section, nous présentons les méthodologies pour construire des systèmes collaboratifs, où nous abordons les systèmes multi-agents (SMA), qui constituent une technologie importante pour la conception des applications distribuées et collaboratives. Dans la deuxième section, nous étudions la propriété de l'interopérabilité, qui constitue une propriété indispensable pour faire communiquer des applications hétérogènes. Plus particulièrement, nous nous intéressons aux architectures-orientées services (SOA), et notamment les services web. Nous expliquons les limitations de cette technologie, et l'importance d'intégrer la notion du web sémantique pour aider à créer des applications plus adaptables et dynamiques. La troisième section introduit la propriété de la malléabilité des collecticiels. Nous présentons l'importance de cette propriété dans la conception des applications collaboratives. Ensuite, nous présentons les approches et les méthodologies pour introduire la malléabilité dans la conception des collecticiels ainsi que quelques exemples de plateformes.

Le troisième chapitre présente le début de notre contribution et il expose les différentes étapes de modélisation de l'architecture logicielle $U3D$ que nous proposons pour la malléabilité des collecticiels. Il est divisé en deux grandes sections. La première section est dédiée à la collaboration Machine-Machine où nous proposons un formalisme de collaboration à base de services web ainsi que le modèle d'architecture logicielle associé. La deuxième section est dédiée à la collaboration Homme-Machine-Homme. Nous proposons un formalisme qui intègre les services web et les agents logiciels afin d'intégrer la propriété de malléabilité dans le modèle d'architecture logiciel que nous présentons également. Ce modèle d'architecture $U3D$ (Universal Directory for Description and Discovery) intègre deux types architectures : Architecture Orientée Service (SOA) et Système Multi-Agent (SMA).

Le quatrième chapitre est encore divisé en deux grandes sections. Dans la première section, nous décrivons les technologies utilisées afin de mettre en œuvre l'architecture

*U3D*. Ensuite, nous proposons une étude en UML pour concevoir les différentes classes du système. Dans la deuxième section, nous proposons une étude de la performance des architectures proposées en termes de nombres de messages émis. Notre but est de comparer les résultats attendues de l'*UDDI4C* et l'*U3D* avec celles du modèle SMA-C (Khezami, 2005).

Le cinquième chapitre présente l'application de l'architecture sur deux projets :

– Le premier domaine est la téléoperation collaborative d'un robot via internet. Notamment, nous nous intéressons à l'extension du système ARITI pour proposer une collaboration plus malléable. Nous présentons principalement la composition de missions de téléopération par des utilisateurs, afin de produire une mission prenant en compte les préférences de chacun d'eux visant à manipuler le robot. L'intégration de services est possible pour fournir des fonctionnalités liées à la téléoperation.

– Le deuxième domaine s'inscrit dans le contexte du projet DIGITAL OCEAN. Nous présentons le projet global, ainsi que la partie où notre travail est situé. Notre but est de concevoir un système collaboratif en ligne, Oceanyd Groupware, afin de collecter des données pour, éventuellement, les charger dans une application 3D hors-ligne, Oceanyd Viewer. L'objectif est d'enrichir un environnement sous-marin à partir des données réelles prises par des plongeurs via le collecticiel. Nous mettons en œuvre deux mécanismes pour satisfaire la malléabilité des systèmes : L'intégration et la composition de services élémentaires. Ainsi, nous tentons de créer une certaine interopérabilité entre Oceanyd Groupware et Oceanyd Viewer, tout en mettant en œuvre des mécanismes pour une meilleure collaboration. Nous proposons une évaluation subjective de l'interface de Oceanyd Groupware, afin de mesurer l'impact de la malléabilité sur la performance des utilisateurs en collaboration.

# Contributions

Nous résumons ci-dessous les principales contributions de cette thèse :

- Etude et classification des modèles d'architectures logicielles pour les collecticiels.
- Exploration de l'intégration des services web avec les agents logiciels, ainsi que les approches pour concevoir la malléabilité des collecticiels (Cheaib et al., 2008b).
- Proposition d'un formalisme pour la collaboration à bases de services web, ainsi qu'un formalisme pour la conception des collecticiels malléables à bases de services web et d'agents logiciels.
- Proposition d'une architecture logicielle pour le support de malléabilité, ainsi qu'une architecture logicielle générique pour les collecticiels malléables (Cheaib et al., 2009) (Cheaib et al., 2008a).
- Application aux projets ARITI pour la téléopération d'un robot (Otmane et al., 2008), et DIGITAL OCEAN (Cheaib et al., 2008c) (Dinis et al., 2008) pour la collecte des données multimédia en ligne.

# Première partie

# ETAT DE L'ART

# Chapitre 1

# MODELES D'ARCHITECTURES ET SYSTEMES COLLABORATIFS

## 1.1 Introduction

Le Travail Collaboratif Assisté par Ordinateur (TCAO), en anglais CSCW (Computer Supported Collaborative Work), mobilise la sociologie, la psychologie, l'informatique voir l'économie pour comprendre le fonctionnement des groupes humains en situation de coopération. Qu'il s'agisse de petits groupes partageant un but commun, nous utilisons le terme "travail coopératif", ou de larges organisations au sein desquels les individus impliqués peuvent présenter des buts conflictuels, nous utilisons alors la notion de "travail collaboratif". Ainsi, La collaboration est un atout essentiel de la société humaine, impliquant une interaction coopérative des personnes en vue d'atteindre des objectifs complexes. L'objectif du TCAO est de trouver les moyens par lesquels les applications collaboratives sont susceptibles d'améliorer la communication entre les individus, tout en mesurant leur impact sur le comportement des groupes (Payet, 2003).

Ce positionnement théorique distingue le TCAO du collecticiel (en anglais, groupware), qui est la concrétisation technologique des directions scientifiques fournies par le TCAO. Ainsi, un collecticiel est un ensemble de technique qui contribue à la réalisation d'un objet commun par plusieurs individus réunis par le temps et l'espace. Autrement dit, il s'agit d'un logiciel permettant à un groupe d'utilisateurs de travailler en collaboration sur un même projet, sans être nécessairement réunis d'une façon synchrone ou asynchrone. D'après une définition proposée par l'AFCET (Association Française pour la Cybernétique Economique et Technique), un collecticiel est un "ensemble de techniques et de méthodes qui contribue à la réalisation d'un objectif commun à plusieurs acteurs, séparés ou réunis par le temps et l'espace, à l'aide de tout dispositif interactif faisant appel à l'informatique, aux télécommunications et aux méthodes de conduite de groupe".

Nous commençons le premier chapitre par une description du concept de collecticiel. Ensuite, nous présentons une étude exhaustive sur les modèles d'architectures des collecticiels.

### 1.1.1 Les Collecticiels

Un collecticiel permet un travail collaboratif et à distance afin de rassembler ainsi des groupes de personnes éloignés sur un projet commun. C'est à la fois un instrument de dialogue, d'apprentissage, de mise en commun et une mémoire collective qui doit être totalement souple en structure et en taille, toujours susceptible d'être modifiée. La Commission générale de terminologie et de néologie [1] a adopté le terme logiciel de groupe de travail (forme abrégée : logiciel de groupe), qui pourraient distinguer les outils de plateformes collaboratives de ceux des plateformes de partage. Les collecticiels sont l'objet de recherches actives depuis plus de vingtaine d'années. A la croisée des interfaces homme-machine, du multimédia et de l'internet, ils possèdent leurs propres conférences internationales. Poussés par le passage à la société de l'information, les collecticiels se banaliseront comme un moyen d'échange de l'information et de travail de groupe (Bass et al., 1992).

En effet, pour (Stiemerling and Cremers, 1998), un collecticiel est un système logiciel déployé dans un environnement composé d'un groupe de personnes, et qui remplit le rôle général de soutien à la collaboration au sein de ce groupe. Les auteurs affirment que les utilisateurs collaborent s'ils travaillent ensemble pour atteindre un objectif commun. L'exemple le plus connu d'un système collaboratif est le courriel qui soutient la communication textuelle entre les membres du groupe, qui est une fonctionnalité utile dans une grande variété d'environnements. Le tableau ci-dessous donne des exemples d'applications considérés comme étant des collecticiels.

Une étude scientifique et rigoureuse des collecticiels est assez difficile, en raison de la nature diverse de deux éléments centraux de l'axe de recherche : les ordinateurs et les êtres humains. D'un côté, les ordinateurs peuvent être décrits et prévus avec précision à l'aide des méthodes formelles. De l'autre côté, certains aspects des groupes qui semblent influencer la manière dont les membres coopèrent sont très complexes et imprévisibles pour être exprimés dans des modèles utiles.

En effet, les chercheurs dans ce domaine se sont souvent basés sur des architectures logicielles à bases de composants (Payet, 2003), (Stiemerling and Cremers, 1998), (Teege, 2000) afin de concevoir des collecticiels qui peuvent être modifier tout au long de leurs utilisations. Ceci facilite essentiellement la réutilisation et l'accélération du processus de développement, ainsi que la capacité d'évolution de leurs structures à la suite de modification, rajout, retrait des composants pendant leurs utilisations.

[1] http ://fr.wikipedia.org/wiki/Groupware

| **Exemple** | Fonctionnalités supportant la coopération |
|---|---|
| Système de téléconférence | Fournit des flux synchrones entre les membres de groupes distribués géographiquement. |
| Chat | Support de communication textuelle synchrone qui prend une forme de dialogue faisant intervenir tout les participants |
| Système de travail partagé | Permet l'utilisation conjointe et asynchrone des artefacts de travail comme des documents textes. Ces systèmes sont souvent renforcés avec les mécanismes de contrôle d'accès, fournissant la sensibilisation aux activités des utilisateurs ou de recherche au sein de l'espace de travail. |
| Système de workflow | Permet un support structuré, asynchrone et coopératif en coordonnant les informations et les flux de documents au sein d'un groupe en fonction de la représentation interne de travaux du groupe. |
| Système d'aide à la décision | Fournit de l'aide pour recueillir des contributions à une discussion, de les organiser et les classer. Il fournit aussi les procédures des votes au sein d'un groupe. |
| Système de planification de groupes | Permet le partage des calendriers individuels ou communs en vue de coordonner les réunions et d'allouer les ressources, comme par exemple les salles de conférences et le matériel. |

TAB. 1.1 – Exemples de quelques fonctionnalités basiques d'un collecticiel (adapté de (Stiemerling and Cremers, 1998))

### 1.1.2 Modèle du Trèfle

Une confusion dans le monde de TCAO prend lieu entre les différentes interprétations des termes collaboration et coopération. Quelques auteurs considèrent ces termes comme des synonymes, alors qu'autres (Dillenbourg et al., 1996) font une distinction entre les deux. Dans notre travail, nous nous appuyons sur le modèle 3C (Ellis, 1994) pour approfondir nos recherches sur le terme collaboration et ses fonctionnalités. Ainsi, nous définissons la collaboration comme étant une activité impliquant la communication, la coordination et la coopération (ou la production, qui est le terme francophone utilisé pour désigner la coopération). Elle peut être synchrone ou asynchrone, au même lieu ou dans des lieux différents. Comme nous pouvons voir dans la figure 1.1, un collecticiel couvre trois fonctions de domaines spécifiques : communication, coordination et production /coopération :

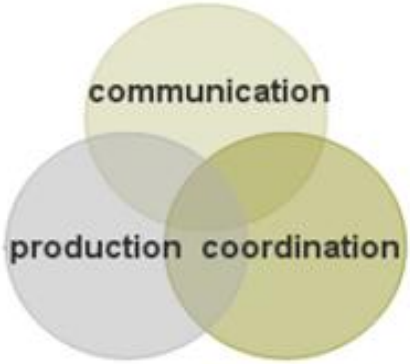



FIG. 1.1 – Le modèle 3C

– L'espace de communication concerne la communication entre les individus. Par exemple, le courriel électronique et les médiaspaces (systèmes conçus pour supporter la communication synchrone par ordinateurs interposés).

– L'espace de coordination codifie les protocoles qui gouvernent la façon dont les tâches sont gérées par les groupes d'utilisateurs. Ces protocoles peuvent être purement sociaux, comme par exemple la règle sociale qui spécifie qu'une personne doit parler à la fois, ou peuvent être formellement spécifiés à travers un système de workflow.

– L'espace de production comprend les artefacts partagés qui sont manipulés de façon coopérative pour réaliser les différentes tâches, comme par exemple les documents ou les données partagées.

## 1.2 Modèles d'architecture pour les collecticiels

Dans cette partie, nous faisons une synthèse des modèles d'architectures logicielles pour les collecticiels. Ces modèles sont étudiés dans (Laurillau, 2002) (dont la plupart des figures dans cette section sont extraites) et (Khezami, 2005). En effet, ces modèles se basent sur des modèles de systèmes interactifs tels Arch (Bass et al., 1992), MVC (Model-View-Controller) (Krasner and Pope, 1988) et PAC-Amodeus (Nigay, 1994), qui s'appuient sur une décomposition fonctionnelle séparant le code du noyau fonctionnel (logique de l'application) de celui de l'interface. Les modèles d'architectures pour les collecticiels étudiés sont organisés selon leurs apparitions chronologiques : ALV, Zipper, CoPAC, PAC*, AMF-C, le méta-modèle d'architecture de Dewan, Clock et DragonFly, Clover, et finalement le modèle C4.

### 1.2.1 ALV

Le modèle d'architecture ALV a donné lieu à la réalisation de la plate-forme Rendez-Vous (Hill et al., 1994), qui se repose sur la notion d'objets partagés. Ainsi, le collecticiel est composé d'un unique noyau fonctionnel partagé pour plusieurs interfaces. Ce modèle possède plusieurs composants (Figure 1.2) :

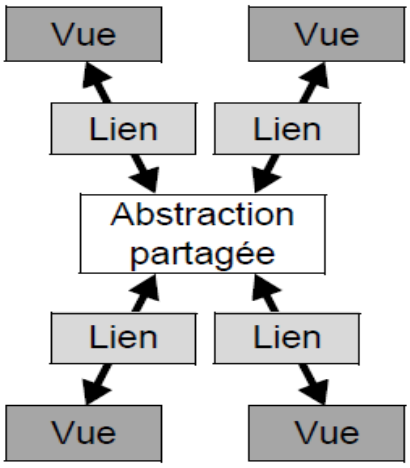



FIG. 1.2 – Le modèle ALV, extrait de (Laurillau, 2002)

– Une abstraction partagée qui représente le noyau du modèle, et qui est partagé par les utilisateurs en collaboration.

– Une vue répliquée qui représente l'interface de chaque utilisateur, et qui interprète les entrées et les sorties.

– Un lien qui fait le lien entre l'abstraction partagée et la vue. En plus, il s'assure que les données de chaque vue sont conformes au données au niveau de l'abstraction.

### 1.2.2 Modèle Zipper

Le modèle Zipper décompose un collecticiel (Figure 1.3) selon quatre niveaux d'états :
– Etat de l'écran qui représente les périphériques d'entrée et de sortie (moniteur, souris, etc.).
– Etat de la vue qui représente les données au niveau de l'interface utilisateur.
– Etat du modèle qui correspond au noyau fonctionnel et aux objets du domaine.
– Etat du fichier qui correspond à la représentation persistante du modèle.

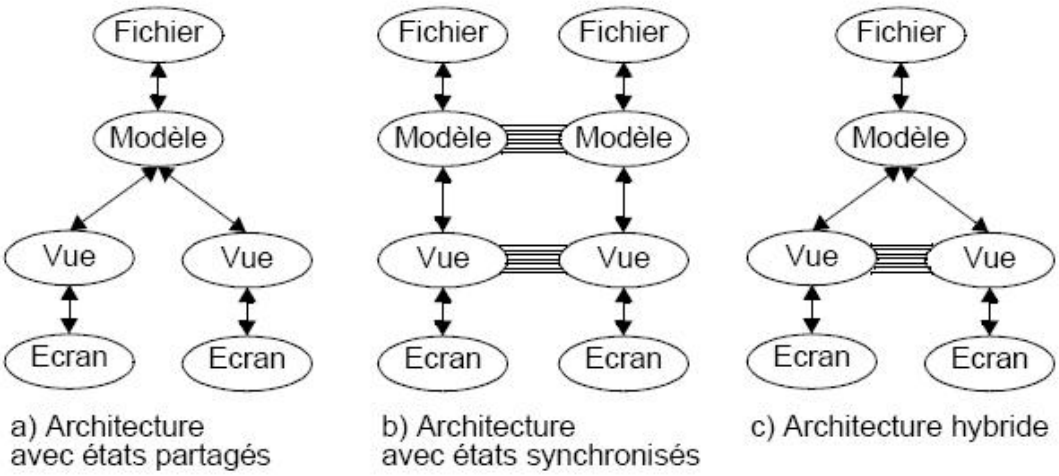



FIG. 1.3 – Le modèle Zipper

Ce modèle ne fait aucune distinction entre les fonctions relevant de la production, de la communication ou de la coordination. En effet, un état partagé est nécessairement alloué à un processus centralisé et un état répliqué est alloué à un processus réparti (Laurillau, 2002). Ceci est confirmé par l'existence du mécanisme de synchronisation qui est chargé de maintenir cohérent un état partagé sur plusieurs processus repartis. En effet, une solution pour obtenir des applications collaboratives est de les concevoir comme des logiciels composables, en associant des composants élémentaires, chacun dédié à la réalisation d'une tâche spécifique. En considérant que les quatre niveaux d'états qui définissent des niveaux d'abstraction sont des composants élémentaires (vue, écran, fichier, etc.), ce modèle satisfait la propriété de la re-configurabilité des systèmes collaboratifs.

### 1.2.3 Modèle CoPAC

CoPAC (Salber, 1995) est une extension du modèle PAC-Amodeus. Dans ce modèle, il existe deux types d'agents : l'agent PAC-Amodeus mono-utilisateur et l'agent collaboratif communiquant avec les autres agents collaboratifs. Ainsi, nous retrouvons quatre facettes :

- Une facette Abstraction (A) : qui gère les concepts du domaine.
- Une facette Présentation (P) : qui est l'interface utilisateur. Elle a pour but d'interpréter les entrées et les sorties.
- Une facette Contrôle (C) : qui fait le lien et le flux de données entre les deux facettes (A) et (P).
- Une facette Communication (Com.) : qui assure la communication entre les agents collaboratifs (Figure 1.4).

La facette Contrôle effectue un lien entre les facettes Abstraction et Présentation vers la facette Communication et réciproquement.

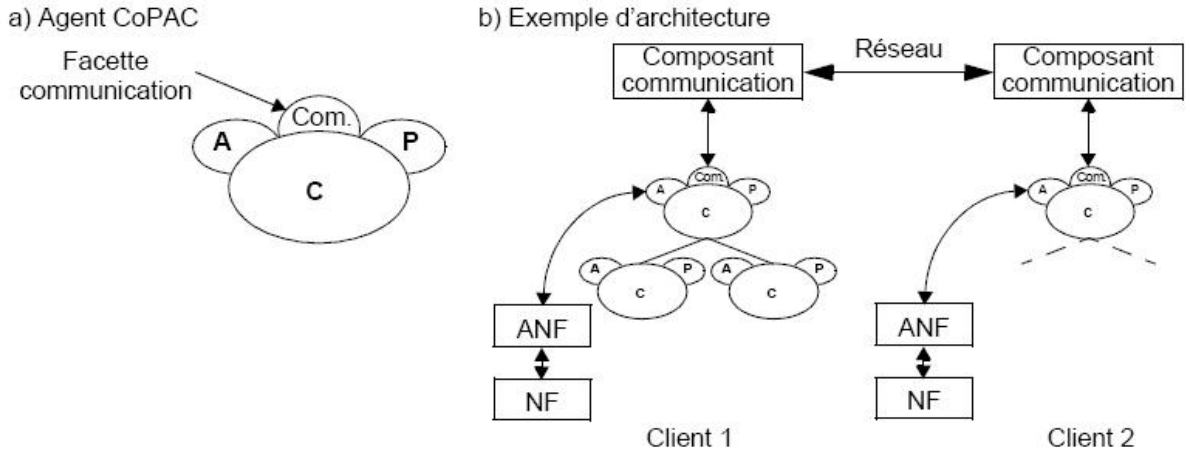



FIG. 1.4 – Le modèle CoPAC

Ce modèle, contrairement à tous les modèles, autorise toutes les combinaisons possibles pour rendre des composants publics ou privés, ce qui rend le système flexible et apte à changer dynamiquement l'état de ces composants (Laurillau, 2002).

### 1.2.4 Modèle PAC*

Ce modèle (Calvary et al., 1997), en plus de la décomposition de ses facettes selon l'agent PAC-Amodeus (Abstraction, Contrôleur et Présentation), il respecte aussi la décomposition de ces facettes selon le modèle du trèfle (3C) : communication, coordination et production. Comme le montre la figure 1.5, un agent PAC* peut exister sous trois formes différentes : centralisée, répartie et hybride. Dans la forme centralisée, cet agent est composé de trois agents reliés à un agent ciment, qui assure la communication entre ces agents et le reste du monde. Dans la forme repartie, les trois agents communiquent directement entre eux et assurent eux-mêmes la liaison avec les autres agents du système. La forme hybride est la combinaison des deux formes précédentes, sachant que l'agent ciment conserve toujours son rôle de lien avec le monde extérieur.

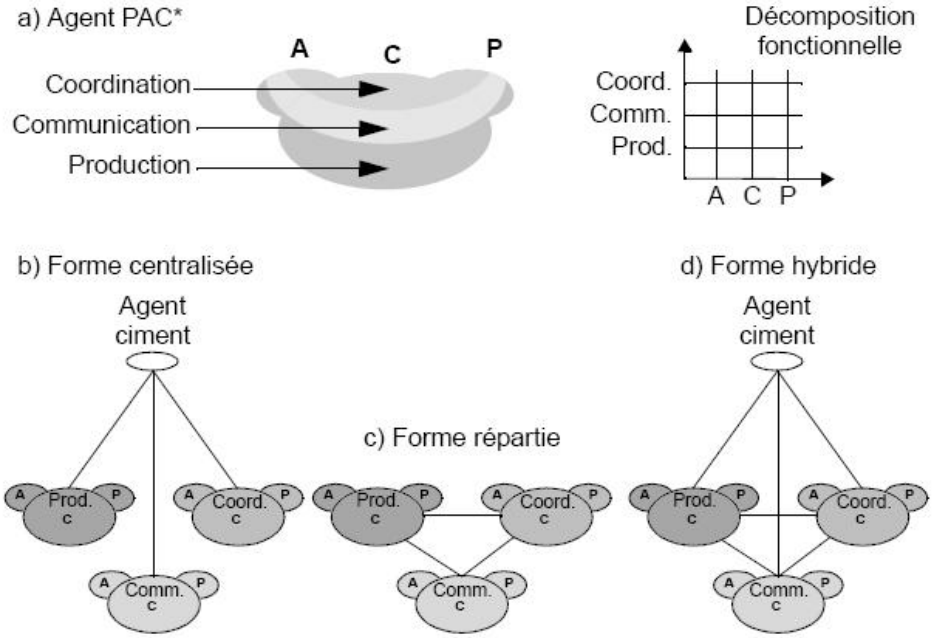



FIG. 1.5 – L'agent Pac*

### 1.2.5 Modèle AMF-C

Le modèle AMF-C (Tarpin-Bernard and David, 1997) repose sur une approche multi-agent multi-facette, qui ne se limite pas aux seules facettes Abstraction, Présentation et Contrôle. Ainsi, ce modèle autorise l'ajout de nouvelles facettes qui s'articulent autour de la facette Contrôle, telles que la facette dédiée au traitement des erreurs, ou la facette de gestion de l'aide.

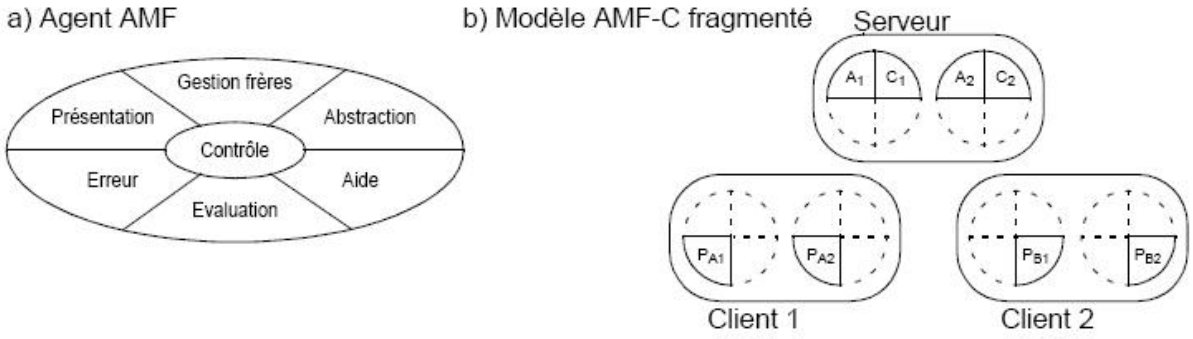



FIG. 1.6 – Le modèle AMF-C

Cette approche se base sur la répartition des agents sur différents sites selon deux stratégies : la première est la fragmentation, qui consiste à fragmenter et à répartir les facettes d'un agent sur différents sites (Figure 1.6). La deuxième stratégie (réplication) considère l'agent complet à l'aide d'un agent local et d'un agent de référence : un agent de référence est partagé par N clients à l'aide de N agents locaux, qui ont la charge de maintenir une vue cohérente de l'agent de référence partagé.

### 1.2.6 Méta modèle de Dewan

Ce méta-modèle (Dewan, 1999) est une généralisation du modèle interactif Arch (Bass et al., 1992), et du modèle Zipper (Patterson, 1995). Selon ce méta-modèle, un collecticiel est constitué d'un nombre variable de couches qui représente plusieurs niveaux d'abstraction (Figure 1.7) : la couche la plus haute (niveau N) est de nature sémantique, et correspond au noyau de l'application, tandis que la couche la plus basse (niveau 0) représente l'interface physique.

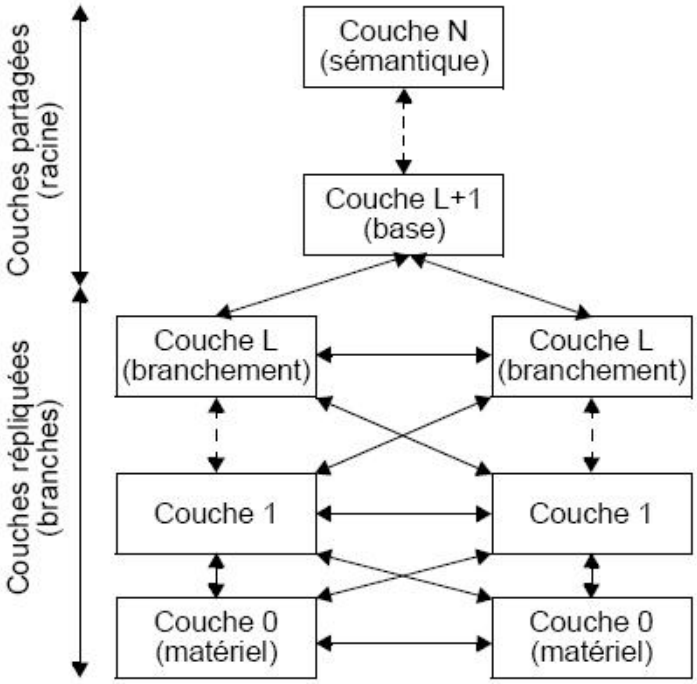



FIG. 1.7 – Le modèle de Dewan

L'architecture globale est constituée d'une racine et de plusieurs branches :

- La racine est composée de couches partagées (niveaux L+1 à N).
- Les branches sont composées de couches répliquées (niveaux 0 à L), reliées à la racine au niveau de la couche L (point d'embranchement).

Les couches communiquent entre elles à l'aide des événements d'interaction qui reflètent l'interaction de l'utilisateur avec le système, ainsi que des événements de collaboration entre les couches appartenant à deux branches différentes. Ce méta-modèle ne décompose pas les couches selon le modèle de trèfle (communication, coordination et production). De plus, il autorise une architecture sans composant public, ce qui limite la collaboration entre les utilisateurs. Ce méta modèle peut être considéré comme un modèle flexible si nous considérons que des couches peuvent être dynamiquement intégrées dans le système.

### 1.2.7 Clock Et DragonFly

Ce modèle d'architecture (Anderson et al., 2000) est un modèle multi-agent qui se base sur le modèle interactif MVC (Krasner and Pope, 1988). Il est constitué de trois facettes :

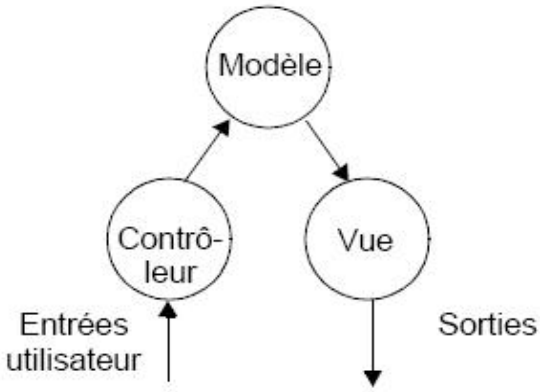



FIG. 1.8 – Le modèle Clock

- `Le modèle` gère les objets du domaine et constitue le noyau du système.
- `Le contrôleur` interprète les interactions de l'utilisateur avec le système.
- `La vue` gère les sorties du système et l'affichage sur l'interface des utilisateurs.

A La différence du l'agent MVC, les facettes contrôleur et vue ne peuvent pas communiquer entre elles, où la communication est monodirectionnelle : le contrôleur indique au modèle les actions réalisées par l'utilisateur, et la vue reçoit les notifications de mise à jour de la facette modèle. Ce modèle ne fait aucune distinction entre les fonctionnalités relevant de la production, de la communication ou de la coordination.

### 1.2.8 Modèle Clover

Ce modèle (Laurillau and Nigay, 2002a) est dérivé du métamodèle d'architecture générique pour les collecticiels de Dewan (Dewan, 1999). Il est dérivé aussi du modèle

de référence Arch (Bass et al., 1992) en appliquant ces cinq niveaux d'abstraction, mais en ajoutant un sixième niveau. Ainsi, le noyau est divisé en deux niveaux : le Trèfle Fonctionnel privé, et le Trèfle Fonctionnel public (Figure 1.9) :

- Le Trèfle Fonctionnel public gère l'ensemble des objets du domaine communs à l'ensemble des utilisateurs, et offre un ensemble de services qui permettent de les manipuler.
- Le Trèfle Fonctionnel privé gère l'ensemble des objets du domaine propre à un chaque utilisateur.

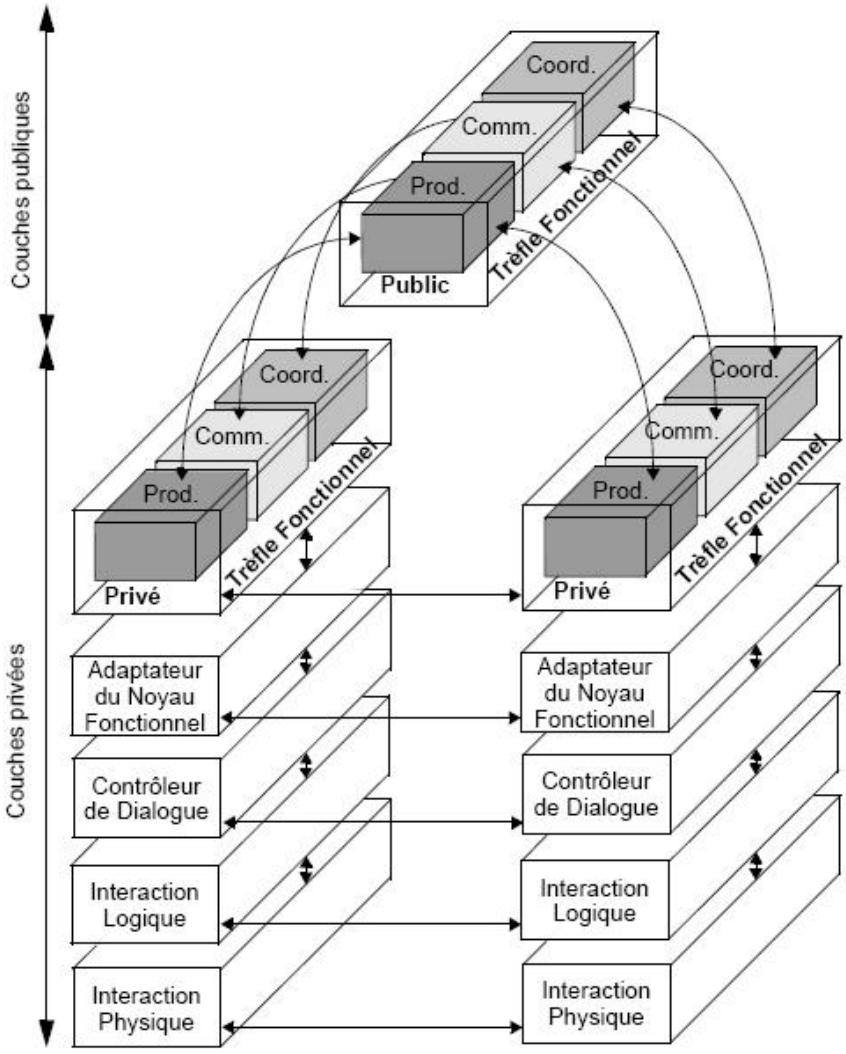



FIG. 1.9 – L'architecture Clover prise de (Laurillau, 2002)

Cette architecture est constituée de niveaux publics et privés : tous les niveaux du modèle sont privés à l'exception du Trèfle Fonctionnel public. La particularité de ce modèle repose sur la structure de son noyau fonctionnel : une abstraction encapsulant trois sous composants dédiés à chacune des trois facettes du modèle 3C : communication, coordination et production. Chaque sous composant offre des services qui relèvent d'une de ces trois classes. En effet, le rôle joué par l'abstraction du Trèfle Fonctionnel est triple :

– L'encapsulation réalisée par cette abstraction agit comme une colle fonctionnelle : elle assure la communication entre les trois sous composants et les autres niveaux dans le modèle.

– Elle gère l'ensemble des objets du domaine communs aux trois sous composants. Cependant, tous les objets ne sont pas identifiés comme relevant d'une des trois catégories définies par le modèle 3C. Au contraire, un objet peut être utilisé à la fois par les trois sous composants.

– Cette abstraction offre un ensemble de services, autres que ceux fournis par les trois sous composants, incluant les services systèmes ainsi que les fonctionnalités relevant de l'action individuelle.

La répartition fonctionnelle de ce modèle établit une trace directe entre les concepts de la conception et la modélisation de l'architecture logicielle. Ce modèle offre une implémentation modulaire qui satisfait les propriétés de la réutilisabilité et l'extensibilité, et donc réduit le coût de développement. En plus, ce modèle ne fixe pas le nombre de couches qui peuvent exister : d'autres couches peuvent être ajoutées pour augmenter la séparation des fonctionnalités et satisfaire la modularité du code, ce qui rend l'architecture logicielle ouverte (Maeda et al., 1997).

### 1.2.9 Modèle d'agent C4

Le modèle C4 est un formalisme d'agent proposé par (Khezami et al., 2005), dédié à la téléoperation collaborative via internet. Il est basé sur le modèle PAC* en mettant en œuvre trois agents, chacun dédié aux trois espaces du modèle 3C. En plus de ces trois espaces, ce modèle met en œuvre un quatrième agent : l'agent collaboration. L'ensemble de ces quatres agents forme un agent collaborateur. Nous pouvons voir une interaction externe entre deux agents collaborateurs dans la figure 1.10 ci-dessous :

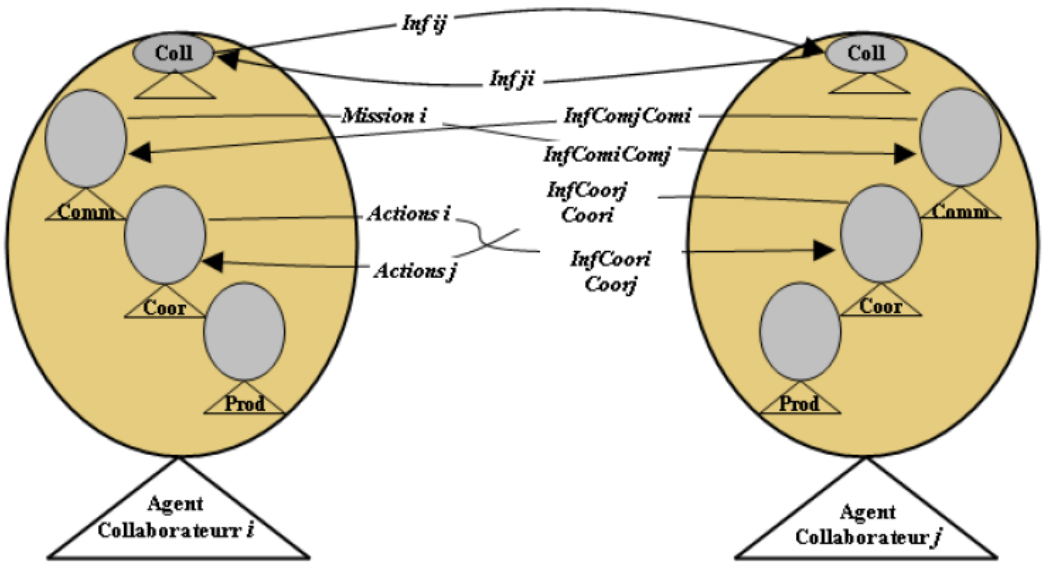



FIG. 1.10 – Interaction durant un processus de collaboration

#### 1.2.9.1 Agent Collaboration

L'agent collaboration assure deux fonctions principales. D'une part, il permet la communication entre les trois agents dédiés de l'agent collaborateur, et d'autre part, il établit une interaction directe entre chaque agent dédié avec son correspondant d'un autre agent collaborateur. Ainsi, ces deux fonctions leurs permet de communiquer ensemble sans l'intervention de l'agent collaboration. Cette communication directe entre les agents dédiés d'un même agent collaborateur, et avec leurs correspondants des autres agents collaborateurs, permet d'augmenter l'efficacité du système. Cela permet d'éviter l'effet du goulot d'étranglement dû à la centralisation.
Les messages de communication de bas niveau (inter-agents) sont échangés entre les agents durant toutes les phases de la collaboration. Par contre, l'agent collaboration assure la communication inter agents dédiés d'un même agent collaborateur qui eux ne communiquent pas directement, et ce pour limiter le nombre de messages échangés, ainsi que pour éviter la perte d'information. En effet, quand l'agent communication finit son travail, il communique ses résultats à l'agent collaboration. Ce dernier se charge d'informer l'agent coordination du commencement de son travail. Idem pour l'agent coordination et pour l'agent production.

#### 1.2.9.2 Agent communication

L'agent communication décide si une mission peut être accomplie ou pas, et c'est donc lui qui décide si la collaboration peut avoir lieu. En conséquence, l'agent communication est la pierre angulaire de l'agent collaborateur. L'agent communication détermine l'état de l'agent en fonction des perceptions qu'il reçoit et de son état antérieur. Cet agent gère toutes les informations qui lui sont communiquées. Ces informations représentent les différentes perceptions qu'il peut recevoir de l'extérieur, comme celles émises par l'agent collaboration. L'agent communication définit son protocole d'interaction avec les autres agents. Ce protocole est choisi selon le type de l'agent destination qui peut être extérieur ou intérieur :

- Agent extérieur : dans ce cas, l'agent communication est en interaction avec l'agent communication d'un autre agent du monde.
- Agent intérieur : dans le deuxième cas, l'agent destination est l'agent collaboration.

#### 1.2.9.3 Agent coordination

L'agent coordination définit toutes les actions que l'agent peut accomplir suite au choix de la mission de l'agent communication. L'agent coordination reçoit un ensemble d'informations qui lui sont envoyées par les autres agents coordination ou par l'agent collaboration. Cet ensemble regroupe les différentes perceptions du monde reçues par ce dernier, ainsi que les informations qui lui sont envoyées par l'agent collaboration.

#### 1.2.9.4 Agent production

L'agent production se charge de l'exécution des actions issues de la collaboration des différents agents du système, qui sont envoyées par l'agent collaboration. Comme pour les deux autres agents, l'ensemble de ses entrées et sorties sont les stimuli qu'il peut produire,

ainsi que l'ensemble des résultats issus de l'exécution des procédures associées aux actions.

L'avantage de ce formalisme est l'utilisation de l'agent collaboration qui, d'une part, assure la liaison entre les différents agents de l'agent collaborateur, et d'autre part avec les autres agents collaborateurs du système. Un inconvénient majeur de cet agent est qu'il est un agent passif, préprogrammé à gérer les agents internes et externes de chaque agent collaborateur du système. L'agent collaboration n'a pas la capacité de rechercher de nouvelles ressources afin d'offrir de nouveaux comportements dans le système.

## 1.3 Quelques systèmes collaboratifs

Les modèles d'architectures présentés dans la section précédente sont reliés et offrent un contexte de travail pour satisfaire la collaboration. Dans ce qui suit, nous présentons quelques systèmes collaboratifs développés dans l'industrie et dans des laboratoires de recherches. Certains de ces systèmes reposent sur les modèles d'architectures conceptuels présentés dans la section précédente.

### 1.3.1 Projet CoVitesse

Les auteurs (Laurillau and Nigay, 2002a) présentent une nouvelle architecture pour la conception de collecticiels, le modèle clover. Ce modèle est une complémentarité de modèles existants en définissant plus explicitement la façon dont les fonctionnalités doivent être structurées à chaque niveau d'abstraction. L'application résultante est illustrée dans la figure 1.11 avec le collecticiel CoVitesse (Laurillau and Nigay, 2002b).

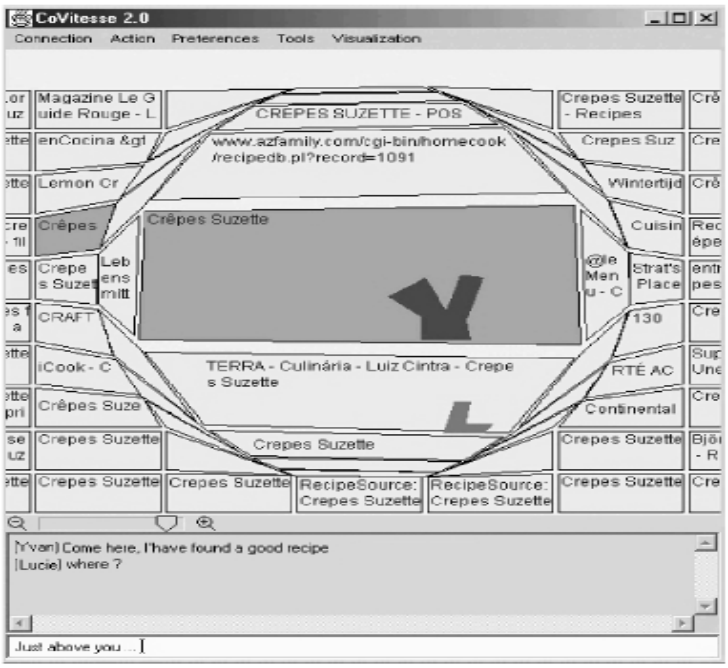



FIG. 1.11 – Capture d'écran du projet CoVitesse

Ce système permet une navigation collaborative sur internet. Les utilisateurs naviguent dans un même espace pour collecter collaborativement de l'information. Ce système

présente l'ensemble des pages affichées sous forme de polygones. Les utilisateurs sont représentés par des formes géométriques colorées. Ainsi, un utilisateur peut naviguer dans l'espace, observer les autres utilisateurs présents, organiser son panier contenant l'ensemble des pages collectées, lire une page en sélectionnant un polygone, communiquer via un forum de discussion situé sous l'espace d'information, ainsi que de créer ou joindre un groupe.

### 1.3.2 Platinum

Pour les auteurs (Ter Hofte et al., 1996), les collecticiels existants sont largement conçus comme des systèmes isolés et propriétaires. Ainsi, un manque de support compréhensif peut se manifester comme un déficit d'intégration. Les systèmes ouverts et modulaires ont le potentiel de fournir la rigidité des solutions isolés et aussi les vertus d'intégration. Les auteurs présentent le modèle conceptuel Co4 (Conférence, Coopération, Conversation et Coordination) qui fournit une vue d'ensemble des fonctionnalités de collecticiel du point de vue de l'utilisateur (Figure 1.12), en distinguant quatre fonctions de base :

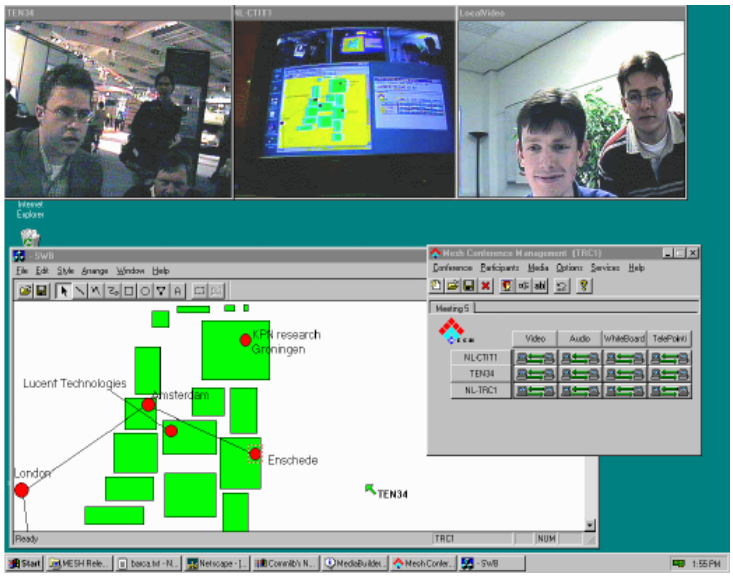



FIG. 1.12 – Capture d'écran du projet Platinum

- Support de conférence : par le biais des fonctionnalités de gestion pour initier, modifier et terminer des conférences. Ces conférences comprennent des membres, des conversations et des politiques de coordination.
- Support de coopération : par le biais du partage des espaces de travail. La coopération contient des fonctions de soutien pour la création, la manipulation et l'observation des objets partagés dans ces. Ainsi, il offre un soutien indirect pour la communication entre tous les membres d'une conférence.
- Support de conversation : par le biais de canaux de conversation, qui fournissent un support de communication directe entre tous les membres d'une conférence. Une conférence peut contenir plusieurs canaux de conversation.
- Support de coordination : par le biais de mécanismes de coordination, qui définissent des politiques que les membres doivent exécutés pour effectuer des actions sur les objets (espace de travail, statues de conversations ou de conférences, etc.).

Ces fonctions permettent de décrire une application collaborative, facilitant la comparaison avec d'autres applications. Ce travail est intégré dans le projet Platinum qui est un grand projet de recherche entre plusieurs institutions.

### 1.3.3 ConversationBuilder

Les auteurs (Kaplan and Carroll, 1992) présentent des investigations sur les environnements qui fournissent un support flexible et actif pour le domaine du TCAO. Le projet ConversationBuilder (CB) (Figure 1.13) essaye de proposer des solutions dans ce domaine à travers des protocoles qui permettent la définition de différents types d'activités, et par les mécanismes des obligations qui permettent d'interconnecter dynamiquement des activités individuelles. Ce projet vise à présenter : une vision théorique pour faire un noyau d'un outil simple pour satisfaire le besoin de la flexibilité et le support actif, le développement d'une architecture ouverte dont le noyau et d'autres outils peuvent être intégrés, et le développement des modèles pour la génération des interfaces dynamiques.

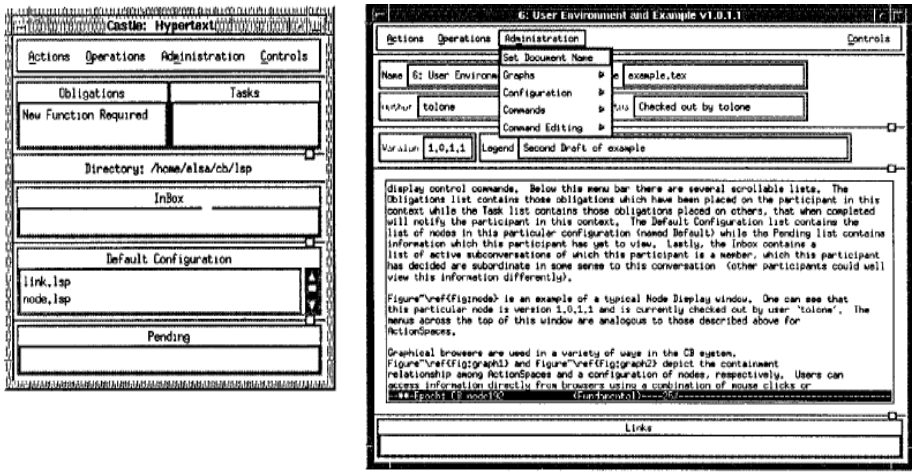



FIG. 1.13 – Capture d'écran du projet ConversationBuilder

### 1.3.4 GroupKit

Les auteurs (Roseman and Greenberg, 1997) ont conçu un ensemble d'outils (toolkit) dédiés aux collecticiels, Groupkit, qui permet aux développeurs de créer des applications pour les téléconférences distribuées et synchrones. L'infrastructure d'exécution du GroupKit se compose d'une variété de processus distribué et disposé sur un certain nombre de machines. La figure 1.14 illustre un exemple de processus en cours d'exécution dans GroupKit, lorsque deux personnes sont en train de communiquer par le biais de deux conférences "A" et "B". Les trois grandes boîtes représentent les trois postes de travail, les ovales sont des instances de processus sur chaque machine, et les flèches indiquent les voies de communication. Les trois types de processus sont :

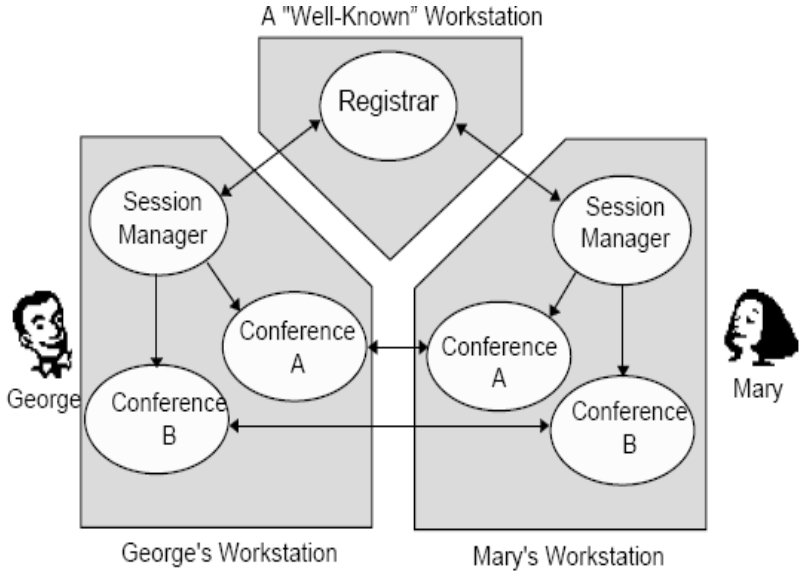



FIG. 1.14 – Illustration de la plateforme GroupKit

– Un composant d'enregistrement (Registrar), qui est le premier processus crée dans une session de Groupkit et le seul processus centralisé dans l'infrastructure d'exécution. Il est doté d'une adresse connue par les autres processus afin d'être facilement trouvé.

– Le gestionnaire de session, qui est répliqué pour chaque participant, et qui permet aux utilisateurs de créer, supprimer, contrôler, participer, ou laisser des conférences. Le gestionnaire de session est à la fois une interface d'utilisateur et une politique qui dicte la manière dont les conférences sont créés ou supprimés, et la façon dont les utilisateurs entrent et sortent des conférences.

– L'application de conférence, qui est une application GroupKit invoquée par l'utilisateur par l'intermédiaire du gestionnaire de session. Ce programme, créé par le développeur de l'application, est un outil collaboratif comme par exemple, un éditeur ou un tableau blanc. Les applications de conférences généralement communiquent ensemble en tant que processus de réplication, dans la mesure où une copie du programme s'exécute sur chaque machine des utilisateurs.

### 1.3.5 TeamWave Workplace

Les auteurs (Roseman and Greenberg, 1997) argumentent que les collecticiels d'aujourd'hui contiennent des trous qui bloquent des interactions sociales et technologiques. Ainsi, ils ne permettent pas aux utilisateurs de se déplacer facilement entre plusieurs modes de travail. C'est pour cette raison qu'ils ont adopté une nouvelle approche, la "métaphore de chambres" ("room metaphor" en anglais) pour la construction des collecticiels. Ils argumentent que les métaphores permettent de travailler ensemble plus aisément. Ils expérimentent cette approche au sein de projet TeamWave Workplace (Figure 1.15), pour justifier comment certains trous de communication peuvent être éliminés. En utilisant cette approche, un utilisateur peut laisser des messages, documents et annotations pour les autres utilisateurs en occupant la même chambre, en même temps.

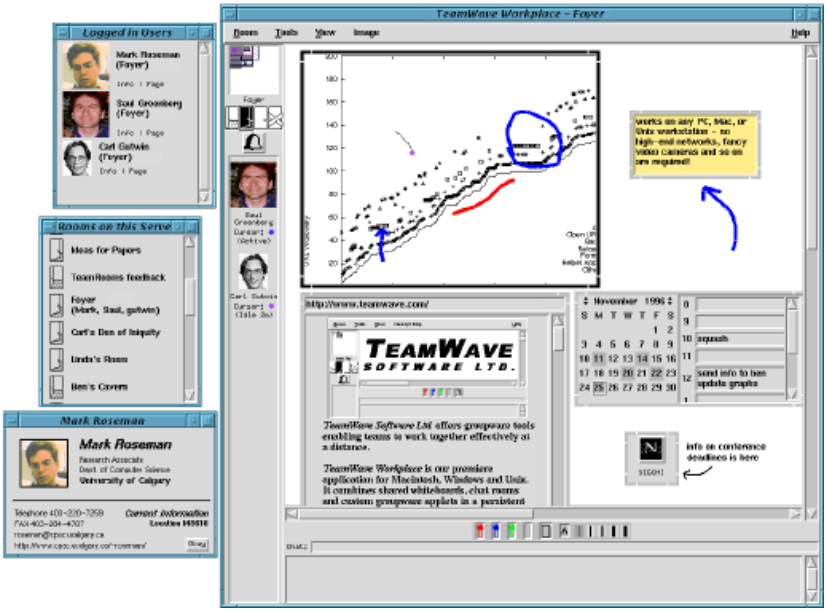



FIG. 1.15 – Capture d'écran de TeamWave Workplace

Dans ce système, un groupe d'utilisateurs géographiquement distribué maintient une ou plusieurs chambres, qui vont servir comme un point focal à leur collaboration. Ces espaces sont localisés sur un serveur et permettent au client de se connecter à travers le réseau. Ils contiennent des applications collaboratives et interactives qui supportent le WYSIWIS (What You See Is What I See) (Stefik et al., 1987). Un des avantages de cette approche est que des chambres individuelles peuvent être personnalisées en ajoutant de nouveaux composants (outils), qui sont aussi des applications collaboratives. Pour l'implémentation, leur solution est basée sur Tc/Tk et Groupkit (Roseman and Greenberg, 1997), qui utilise une hiérarchie de fenêtres partagées.

### 1.3.6 Disciple

Le projet Disciple (Marsic, 1999) (DIstributed System for Collaborative Information Processing and LEarning) présente un cadre pour le partage des applications à base de "JavaBeans" en temps réel, dans des environnements collaboratifs synchrones. Un composant générique de collaboration met en œuvre un environnement "plug-and-play" qui permet la collaboration avec des applications qui ne sont pas particulièrement collaboratives. Le système Disciple définit une architecture client-serveur, qui vise à offrir à la fois une boîte à outils pour le développement des applications collaboratives à des fins spéciales, ainsi qu'un cadre pour le partage des applications mono-utilisateur.
En effet, le système définit un cadre qui décrit l'architecture logicielle en couches, du côté du client, avec une interface multimodale homme/machine situé au-dessus de la couche applets/beans de l'application. Cette dernière utilise la couche de la collaboration qui est la couche la plus inférieure du système. Couvrant tous les niveaux du système, des agents intelligents fournissent les mécanismes de gestion. Cette initiative peut être classée comme une composante collaborative axée sur l'extensibilité du programmeur. Nous pouvons voir une capture d'écran du système dans la figure 1.16 :

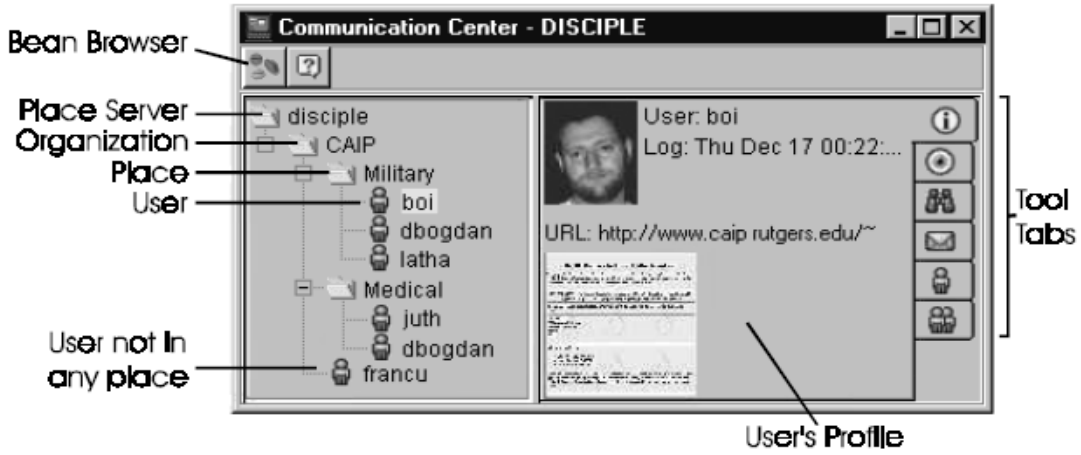



FIG. 1.16 – Capture d'écran de la platforme Disciple

### 1.3.7 JViews

JViews (Grundy and Hosking, 2002) est une boîte à outils pour la construction des composants de vue, ainsi que des répertoires de dépôts pour les systèmes à multi-vue. Il met en œuvre un chevauchement de vues sur des informations partagées qui correspondent à des utilisateurs différents. La cohérence entre les différents points de vue est mise à jour automatiquement par la propagation des modifications à un répertoire partagé, puis à tous les vues dans le système selon le modèle collaboratif de MVC (Krasner and Pope, 1988). En effet, JViews définit les relations entre les composants qui sont utilisés pour les structures de données, ainsi que pour agréger et maintenir la cohérence de ces composants. Finalement, JViews peut être classé comme un élément d'une architecture collaborative en mettant l'accent sur la composition de point de vue programmeur.

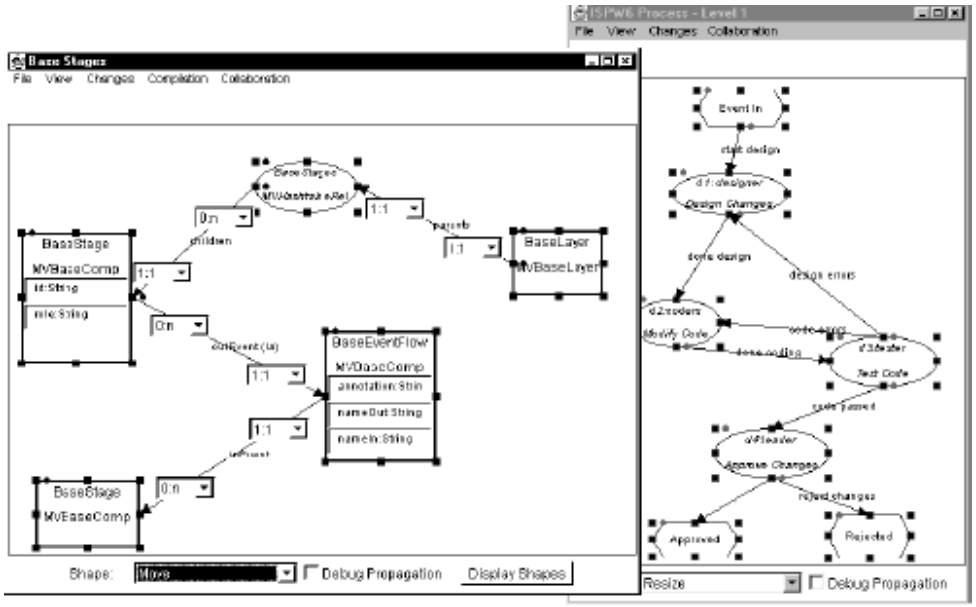


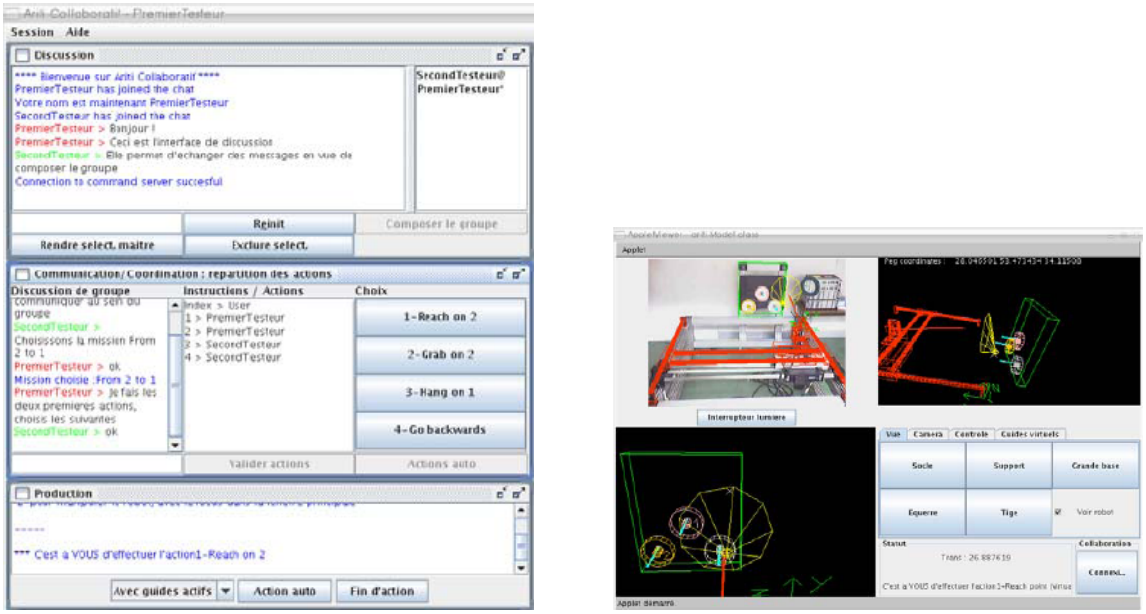

FIG. 1.17 – Capture d'écran de la plateforme Jviews

### 1.3.8 ARITI-C

Le projet ARITI [2] (Otmane et al., 2000) est initié par Samir Otmane pendant sa thèse en 2000 (Otmane, 2000). Les premiers travaux ont permis la mise en place du premier système de téléopération de robot en réalité augmentée via Internet en France. ARITI a été mis en ligne pour la première fois sur le site web du Laboratoire en 1998. Depuis Octobre 1999, il est référencé par la "NASA Space Telerobotics Program"[3]. Nous avons présenté le projet ARITI plus en détails dans le chapitre 5.

La dernière version d'ARITI, mise en ligne en 2005, intègre des fonctionnalités de télétravail collaboratif mises en place par Narjes Khézami pendant sa thèse (Khezami, 2005). Ainsi, le système ARITI-C met en place un système multi-agents pour la collaboration. Les utilisateurs communiquent, coordonnent et finalement passent à la production en partageant un robot virtuel et un robot réel. Ce système est basé sur le modèle C4 présenté dans la section précédente.

Nous pouvons voir une capture d'écran du collecticiel ARITI-C dans la figure 1.18 :




(a) Interface de collaboration

(b) Interface de téléoperation

FIG. 1.18 – Captures d'écrans des interfaces d'ARITI-C

## 1.4 Bilan et Analyse

Pour (Khezami et al., 2005), une architecture logicielle ne peut pas être bonne ou mauvaise. Cependant, sa qualité se juge suivant les attentes de l'utilisateur final du logiciel, ainsi que des qualités de ce dernier qui doivent être définies par le concepteur. Ainsi, une architecture logicielle conforme aux attentes du travail de groupe doit mettre en évidence

[2] $http://ariti.ibisc.univ-evry.fr$
[3] $http://ranier.oact.hq.nasa.gov/telerobotics_page/realrobots.html$

les éléments suivants :

- Bien distinguer les actions individuelles des actions collectives, ainsi que de classer les actions collectives selon l'un ou l'ensemble des trois espaces fonctionnels (production, communication et coordination).

- Bien identifier les types des composants qui gèrent les données, c'est-à-dire, les composants privés pour les données (ressources) privées, et les composants publics pour les données collectives.

- Prendre en compte l'observabilité des actions et des données, c'est-à-dire, si un utilisateur peut voir ce que font tous les participants ou pas, et si oui, comment il peut les voir.

Pour (Laurillau and Nigay, 2002a), une architecture logicielle est assimilée à un ensemble organisé de composants, dont les interactions sont médiatisées par des entités spécialisées ou connecteurs. Le processus de conception d'une architecture logicielle recouvre deux activités principales :

– La définition de la décomposition fonctionnelle du système, qui consiste à identifier les fonctionnalités regroupées sous forme d'entités logicielles ou de composants.

– Le choix d'une organisation structurelle (vue statique et dynamique), où il s'agit de choisir un schéma d'organisation de différents composants identifiés lors de la décomposition fonctionnelle, ainsi qu'un protocole d'échanges entre ces composants.

Pour (Payet, 2003), l'imprévisibilité des besoins qui vont émerger au cours du travail collaboratif, fait qu'il est nécessaire que les systèmes logiciels puissent supporter un travail coopératif sensible au contexte dans lequel ils seront utilisés. Ce travail s'exprime par la situation coopérative qui se présente (organisation, procédures, style de coopération), la spécificité de la tâche à réaliser, et les paramètres dans lesquels la coopération prend place (distribution géographique, base logicielle et matérielle, connections réseaux). Afin de supporter cela, l'auteur affirme qu'un collecticiel doit posséder un mécanisme d'incorporation accessible aux utilisateurs, et doit aider les utilisateurs à interpréter ce mécanisme et les assister dans l'adaptation de l'application à la situation en cours. Le système doit être suffisamment `adaptable` pour pouvoir supporter de nouvelles façons d'utilisation, façons non prévues par le développeur. Les auteurs affirment aussi que les exigences des utilisateurs ne se limitent pas seulement à des préférences de l'apparence de l'interface graphique de l'outil, mais également à des préférences au niveau des sous fonctions de l'application, qui concernent plutôt les besoins des utilisateurs en termes de `service` rendu par l'outil.

### 1.4.1 Nouvelles exigences

A partir de notre étude sur les modèles d'architectures les plus pertinents dans la littérature, nous procédons à la classification de ces modèles selon des critères d'ouverture, d'adaptabilité et d'interopérabilité, qui vont éventuellement converger vers la propriété de la malléabilité. Pour l'instant, nous définissions ces termes comme suit :

- `Ouverture` : L'un des mécanismes le plus important dans la conception des collecticiels est l'extension des `services` offerts en cours de son exécution. C'est aussi de renchérir que les systèmes collaboratifs doivent être apte à intégrer de nouveaux modules, en même temps que l'environnement de travail subit des variations de contexte. Ainsi, le système doit supporter la réutilisation et la combinaison d'outils pour des différentes catégories d'équipes. Au moment où leurs besoins changent, les membres de l'équipe doivent être en mesure d'étendre et de modéliser leur environnement de travail de sorte à ce qu'il soit satisfaisant à leurs besoins. Ainsi, il doit pouvoir être `ouvert`, et supporte `l'intégration` de nouveaux outils dynamiquement à tout moment, ainsi que d'être capable à répondre à des besoins d'évolution. Ce dernier point est d'autant plus problématique que rien ne permet d'anticiper la nature de ces futurs outils, il n'est pas possible, en effet, de prévoir quels seront les outils adéquats dans la collaboration de demain.

- `Adaptabilité` : Nous définissons un système adaptable comme un système qui change la structure de ses composants selon la tâche, ou le contexte dans lequel les utilisateurs sont en train de collaborer. Ainsi, le but est de `composer` les modules du système afin de générer de nouveaux services, qui peuvent satisfaire les préférences des utilisateurs sans arrêter son exécution, ni manuellement coder et recompiler, ainsi réduire le coût de développement et gagner plus d'efficacité.

- `Interopérabilité`[4] : Selon la définition classique, c'est la capacité que possède un système à fonctionner avec d'autres systèmes sans restriction d'accès ou de mise en œuvre. Ainsi, l'interopérabilité est considérée comme très importante dans de nombreux domaines, dont l'informatique, le médical, l'électrotechnique, l'aérospatiale, le domaine militaire et l'industrie en général. Compte tenu du fait que certains éléments utilisés par ces systèmes sont produits par des constructeurs divers, avec des méthodes variées, et qu'ils répondent à des besoins spécifiques, l'idée consiste à définir une norme ou un ensemble de normes que chaque élément peut intégrer dans son propre fonctionnement. Cette norme joue un double rôle : elle est un indicateur de la façon dont le dialogue entre les différents systèmes doit s'opérer et cristallise les besoins de ce dialogue ; elle est ensuite une passerelle de communication, qui va pouvoir éventuellement s'adapter aux besoins changeants de ces systèmes.

Les modèles d'architectures pour les collecticiels présentés dans ce chapitre ont été déjà classifiés dans la littérature selon divers propriétés, comme l'observabilité des ressources (Laurillau and Nigay, 2002a) (Khezami et al., 2005), mais aucun travail n'a essayé de classifier ces modèles selon les propriétés citées définis dans cette section, afin de répondre plus efficacement aux besoins évolutifs des utilisateurs en collaboration.

[4]http ://fr.wikipedia.org/wiki/Interopérabilité

| Modèles d'architecture | Ouverture | Adaptabilité | Interopérabilité | Flexbilité |
|---|---|---|---|---|
| ALV | - | - | + | Partielle |
| Zipper | - | + | - | Partielle |
| CoPAC | - | + | + | Partielle |
| PAC* | - | + | + | Partielle |
| AMF-C | - | + | + | Partielle |
| Dewan | - | + | + | Partielle |
| Clock et DragonFly | - | - | - | Aucune |
| Clover | - | + | + | Partielle |
| C4 | - | - | + | Partielle |

TAB. 1.2 – Classification des modèles d'architectures

Ainsi, nous classons les modèles d'architectures des collecticiels selon la propriété de la flexibilité (qui regroupe les trois propriétés d'ouverture, d'adaptabilité et d'interopérabilité). Cette propriété se traduira dans les chapitres suivants par la propriété de la malléabilité des collecticiels. Nous remarquons qu'aucun modèle d'architecture ne satisfait pleinement cette propriété. Notre objectif est de proposer un modèle d'architecture pour les collecticiels qui est apte la satisfaire.

## 1.5 Conclusion

Dans ce chapitre, nous avons présenté des modèles d'architectures logicielles pour les collecticiels, ainsi que quelques exemples de systèmes développés à partir de ces modèles. Ainsi, l'objectif du TCAO est de trouver des moyens pour que les logiciels améliorent la collaboration entre les individus. En effet, l'invention du collecticiel est un défi, puisque la nature de la collaboration continue à changer en conséquence de l'évolution des besoins, mais aussi comme une conséquence de la façon dont les systèmes eux-mêmes ont tendance à changer les relations de travail. En conséquence, les auteurs dans le domaine affirment que les systèmes doivent s'adapter pour refléter le caractère imprévisible des différences entre les exigences du travail collaboratif lors de l'analyse et les besoins réels. A partir de notre étude, nous avons réalisé que les modèles présentés sont souvent statiques : ils ne sont pas conçus d'une manière à générer de nouveaux composants lors de l'exécution du système. Dans ce qui suit, nous présentons quelques outils et concepts qui nous aidons à présenter un modèle d'architecture pour les collecticiels à bases de services web et d'agents logiciels.

# Chapitre 2

# INTEROPERABILITE ET MALLEABLITE DES SYSTEMES COLLABORATIFS

Dans ce chapitre, nous abordons deux thématiques différentes. En premier lieu, nous présentons une étude sur les agents et systèmes multi-agents, qui sont utilisés pour la construction des systèmes collaboratifs. Nous présentons ensuite les services web, ainsi que les protocoles associés afin d'interopérer la communication entre différentes plateformes hétérogènes. Nous faisons une comparaison entre les deux technologies citées afin d'en tirer les avantages de chacune, ainsi que la synergie de leur intégration. La deuxième section introduit la notion de la malléabilité des collecticiels. Nous présentons ce concept à partir des définitions qui se trouvent dans la littérature. Puis, nous citons diverses approches qui visent à introduire la malléabilité dans les systèmes collaboratifs. Finalement, nous citons quelques modèles de collecticiels malléables. Cette étude nous conduit à notre contribution générale, qui est la conception d'un modèle d'architecture générique et malléable pour les collecticiels.

## 2.1 Agents et Systèmes Multi-Agents (SMA)

Les architectures logicielles doivent supporter les exigences fondamentales pour la coopération distribuée : partage d'information, mise à jour constant et accès partagé à un ensemble de services, etc. Les approches et technologies pour supporter cette nouvelle façon de travail sont encore en cours de recherche, empruntant des concepts de différents domaines, tels les systèmes du workflow, les systèmes collaboratifs, les systèmes basés sur les événements, etc. Dans le passé, les utilisateurs ont principalement collaboré entre eux à travers l'échange des informations. Avec le début des technologies comme le web sémantique, les architectures orientées-services (SOA) et les améliorations de la bande passante, les collaborateurs se sont rendu compte de la flexibilité de collaborer par l'échange de services universellement accessibles. Cependant, cette flexibilité a provoquée l'augmentation de la complexité des systèmes. L'autonomie et l'intelligence des agents logiciels ont fortement augmenté l'automatisation dans beaucoup de domaines opérationnels. Un

avantage majeur de l'utilisation des agents est leur capacité à aider les êtres humains en collaboration, avec des mécanismes logiciels.

Dans la littérature, plusieurs systèmes de collaboration ont été développés, mais ils ont tous un inconvénient d'offrir une architecture fermée et souvent spécifique à chaque type d'application. Nous donnons ici une vue d'ensemble sur les recherches faites dans le domaine des systèmes multi-agents appliqués au TCAO. Nous présentons aussi quelques modèles de base des systèmes collaboratifs conçus à partir de ces technologies.

### 2.1.1 Définition

Plusieurs définitions de l'agent logiciel existent dans la littérature. En effet, (Khezami et al., 2005) a défini l'agent comme un objet informatique (au sens des langages objet) dont le comportement peut être décrit par un script qui dispose de ses propres moyens de calcul, et qui peut se déplacer pour communiquer avec d'autres agents. Un agent doit nécessairement être motivé par un but à atteindre, sinon son existence dans son environnement n'aurait pas de sens. Il peut percevoir l'environnement mais peut n'en posséder qu'une représentation partielle, et parfois même aucune. Il peut communiquer avec les autres agents de son environnement et doit avoir des compétences qui lui permettent d'atteindre ses objectifs. Selon (Maamar et al., 2003), un agent est un morceau de logiciel qui agit d'une façon autonome pour entreprendre des charges au nom des utilisateurs. Les auteurs affirment que la conception de beaucoup d'agents logiciels est basée sur le fait que les utilisateurs doivent seulement indiquer un but à niveau élevé au lieu de publier des instructions explicites, et laissant les décisions à l'agent. L'agent montre un certain nombre de dispositifs qui le rendent différent des autres composants traditionnels. Les principales caractéristiques des agents sont (Khezami et al., 2005) (Ferber, 1995) :

- La modularité : Interconnecter et rendre accessibles en tout lieu et à tout moment des ressources informatiques existantes (données, informations, traitements), intégrer des systèmes existants, de les faire coopérer, interagir et collaborer avec un ou plusieurs utilisateurs.
- L'autonomie : Un agent autonome est capable de prendre des décisions selon des critères qui lui sont propres, et sans l'intervention d'un autre agent ou d'un opérateur humain.
- L'interaction : Un agent peut agir sur le monde, c'est à dire sur les autres agents présents dans son univers et sur l'environnement lui-même. Cette intervention peut prendre la forme d'une modification de l'état des autres agents qu'il côtoie, que ce soit au niveau de leurs connaissances ou au niveau de leur activité.
- La coordination : Un système multi-agent est une collection de points de vue locaux différents, voir conflictuels, qui établissent un ou plusieurs réseaux de relations entre les agents qui sont complexes et multiples, et qui peuvent porter sur des buts, actions ou des ressources. Cette coordination permet de répondre aux différents avantages, notamment la résolution de conflits.
- La personnalisation : Un SMA doit adapter continuellement sa relation avec l'utilisateur, ce qui conduit à une personnalisation des comportements. Cette fonctionnalité permet de maintenir des informations sur les préférences de l'utilisateur et peut être construite explicitement ou déduite par le système en observant son comportement.

– L'intelligence : L'intelligence d'un agent est basée sur un agrégat de caractéristiques et concerne la capacité à apprendre, à communiquer et être autonome.
– L'émergence : Des éléments perturbateurs (l'environnement, l'utilisateur ou d'autres agents) imposent au système de s'adapter au contexte et de se restructurer en permanence afin de conserver des performances acceptables.

### 2.1.2 Les Systèmes multi-agents (SMA)

Les SMA permettent de répartir un problème sur un certain nombre d'entités coopérantes. Ces entités ou agents sont autonomes et interagissent dans un environnement pour la résolution de problème qui dépasse les capacités individuelles de chaque agent. En fonction de la taille d'un agent, de sa complexité, de ses connaissances et de son raisonnement, nous pouvons classer les approches multi-agents en trois grandes catégories : cognitive, réactive et hybride (Khezami, 2005).

**L'approche cognitive :** Les concepteurs des SMA cognitifs s'inspirent du comportement humain pour définir la structure et le raisonnement d'un agent cognitif. Pour pouvoir anticiper ou expliquer ses actions et celles des autres agents, un agent cognitif doit posséder des connaissances sur les autres agents. Généralement, les systèmes multi-agents cognitifs sont composés d'un petit nombre d'agents de grande granularité, i.e., ils possèdent un minimum de connaissances et des comportements de haut niveau leur permettant de s'organiser, de se regrouper, de coopérer, d'apprendre à coopérer en se servant de leurs expériences, et également de prévoir les résultats de leurs comportements.

**L'approche réactive :** Les concepteurs de cette approche se sont inspirés des phénomènes biologiques, et tentent de les reproduire par la suite. Les agents sont dotés d'un modèle de comportement du style automate selon un modèle stimulus-réponse. Contrairement aux agents cognitifs, les sociétés d'agents réactifs sont composées d'un nombre considérable d'agent de faible granularité. En général, un agent réactif ignore ses expériences passées car il ne possède pas de processus de raisonnement sophistiqués qui lui permettent de planifier ou d'apprendre. Ces agents sont très rapides dans la prise de décision car ils sont dotés d'un minimum de connaissances et de modèles de raisonnement à base de règles. Pour être fidèle aux modèles biologiques, les concepteurs d'agents réactifs évitent d'utiliser toute communication directe entre agents. Les agents ont alors recours à l'observation des changements qui affectent l'environnement à travers ses différents états, ce qui assure une transmission indirecte des connaissances entre agents.

**L'approche hybride :** Pour surmonter les faiblesses de chacune des deux approches précédentes (la difficulté de mise en œuvre de l'approche cognitive et le manque de modèles formels dans l'approche réactive), plusieurs chercheurs se sont intéressés à l'approche hybride (Khezami et al., 2005). Le postulat de cette approche est de maintenir une certaine réactivité de l'agent en le dotant de composants réactifs et de rendre les autres composants cognitifs pour garantir un raisonnement de qualité. Les fondateurs de cette approche argumentent son intérêt par les avantages qu'elle procure, notamment :

- un agent hybride a une structure modulaire, ce qui est concrètement recommandé dans le développement de tout processus artificiel pour garantir l'évolution et la maintenance du système.

- les capacités de traitement d'un agent peuvent être améliorées car ces différents composants peuvent fonctionner simultanément.
- le composant réactif de l'agent devient plus performant car l'organisation des connaissances d'un agent en partitions permet à chacun des composants réactifs ou cognitifs de manipuler partiellement ou totalement cette connaissance.

### 2.1.3 Quelques Travaux

Nous présentons brièvement quelques modèles multi-agents dédiés pour la collaboration (Khezami et al., 2005).

– (Cabri et al., 2004) présente une approche pour la communication basée sur l'utilisation des agents mobiles, où un message devient une entité active, qui permet le filtrage du contenu et qui laisse le message lui-même décider pro-activement de ce qu'il doit faire avec son contenu, et comment le montrer à l'utilisateur final. Cette approche peut être intégrée dans des applications de communication existantes, ce qui facilite son exploitation par les différents utilisateurs, et permet aussi d'unifier plusieurs systèmes de messages en un seul.

– (Selliah et al., 2004) présente les travaux sur la collaboration des groupes assistée par des agents, où ils introduisent une plateforme appelée Eksarva, qui supporte la modélisation formelle de la collaboration comme une fonction de comportement et des règles du workflow. Cette approche de modélisation est utilisée afin de programmer des agents qui exécutent des activités intégrées, de gestion de la connaissance nécessaire pour une collaboration efficace du groupe.

– (Frey et al., 2003) propose l'utilisation d'un système multi-agent pour la gestion de la chaîne de provision en respectant quelques perspectives comme l'organisation, le contrôle, et l'exécution. Cette approche considère des rapports non fonctionnels tels que la flexibilité et la précision. Elle incorpore l'utilisation des techniques de modélisation de l'industrie standard qui utilisent le langage UML.

– (Cabri et al., 2003) propose une application basée sur un agent mobile. Elle est exploitée pour étudier les différents types d'interactions d'agent. Ces interactions sont gérées séparément de la logique de l'agent au moyen de rôles. Cette approche permet le développement des applications basées sur des agents flexibles pour exécuter des tâches automatiques.

### 2.1.4 Bilan

Les modèles d'agents collaboratifs présentés dans ce chapitre, ne présentent que des normes et des modèles spécifiques à leurs problématiques. La collaboration dans ces modèles consiste à une aide fournit pour l'utilisateur, ainsi qu'entre les différentes entités du système. En plus, ces applications offrent des systèmes d'automatisation globale qui ne considèrent que la coordination entre les agents au sein d'un SMA. Dans notre travail, nous considérons la collaboration, d'une part, comme étant une coopération entre les membres d'une équipe et, d'autre part, une réalisation d'un produit fini. La coopération entre les membres d'une équipe est déterminée par une communication et une coordina-

tion entre tous ces membres. Ainsi, la collaboration est définie par la communication, la coordination et la production.

Selon (Svirskas et al., 2005), les agents logiciels et le web sont la base pour l'automatisation du processus de la collaboration. Alors que la technologie d'agent développe des systèmes dans lesquels des agents autonomes y résident, et qui répondent à leurs objectifs en interagissant dans un esprit de collaboration, le web offre une vaste infrastructure d'information et de communication. Dans la section suivante, nous introduisons la propriété d'interopérabilité que les services web peuvent offrir, afin d'améliorer le processus de collaboration.

## 2.2 Services web

Les utilisateurs essayent d'avoir un accès pragmatique à des "services logiciels", même en étant géographiquement distribués et en utilisant des applications différentes. Le support des services web a pour but d'offrir une interopérabilité entre plusieurs systèmes collaboratifs (Jerstad et al., 2005). L'architecture orientée-services (SOA) considère un modèle de composants intégré, où les interfaces sont vues comme étant des applications modulaires décrites avec un langage standardisé. Ainsi, en encourageant la réutilisation des services basiques, l'architecture orientée-service va pouvoir offrir des solutions efficaces et flexibles.

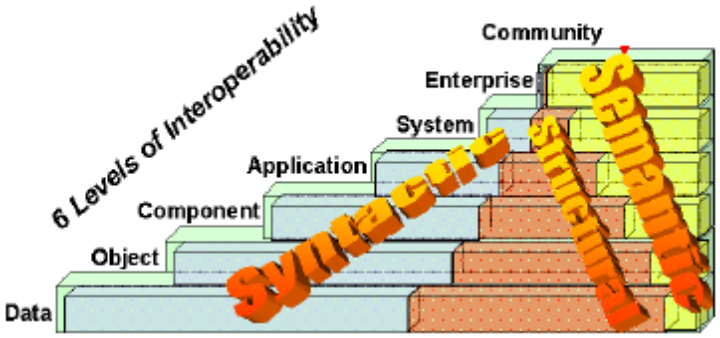



FIG. 2.1 – Les dimensions d'intéropérabilité et d'intégration extrait de (Obrst, 2003)

En effet, le terme interopérabilité est lui-même susceptible à plusieurs définitions. (Wright, 2005) définit le terme comme suit : *"L'interopérabilité permet la communication, l'échange de données, ou l'exécution d'un programme entre divers systèmes d'une manière qui oblige l'utilisateur à avoir peu, ou pas de prise de conscience des opérations sous-jacentes de ces systèmes. C'est le but ultime de la construction d'une solution aujourd'hui."* Les auteurs se basent sur la définition de (Guest, 2005) qui distingue l'interopérabilité (permettant la communication et l'échange de données entre différentes plateformes), de la notion de migration (réécrire un composant pour fonctionner sur une autre plateforme), et la portabilité (le déplacement d'un élément venant d'un autre fournisseur, mais utilisant la même plateforme).

Les auteurs (Obrst, 2003) distinguent l'interopérabilité qui est utilisée dans le contexte des systèmes, de la notion d'intégration qui est utilisée dans le contexte des données :

l'intégration est décrite en fonction du degré de la correspondance syntaxique, structurelle et sémantique entre les deux sources de données. Par contre, l'interopérabilité implique une interopérabilité entre les composants d'un logiciel, en termes de niveau d'agrégations des composants. Dans la figure 2.1 extraite de (Obrst, 2003), les auteurs montrent les six niveaux de l'interopérabilité contre les trois types d'intégration. En effet, l'objectif des services web est de faciliter l'accès aux applications entre les organismes et les clients, ainsi que de simplifier les échanges de données. Ils poursuivent un but de l'informatique distribuée où les applications pourraient être interopées à travers le réseau, indépendamment de leurs plateformes et de leurs langages d'implémentation. Dans ce sens, ils s'inscrivent dans la continuité de travaux tels que CORBA[1] (Common Object Request Broker Architecture), en apportant toutefois une réponse plus simple, s'appuyant sur des technologies et standards reconnus dans l'industrie.

### 2.2.1 Quelques définitions et standards

Le W3C [2] définit un service web comme : *"un système logiciel qui agit comme un support interopérable dans l'interaction machine-machine. Ce système possède une interface décrite selon un format traité et compris par la machine (spécifiquement WSDL). Les autres systèmes interagissent avec le service web selon sa description prescrite en utilisant des messages SOAP (Simple Object Access Protocol), typiquement transporté en http avec une sérialisation XML en conjonction avec d'autres normes du web"*. En d'autres termes, un service web est un composant implémenté dans n'importe quel langage, déployé sur n'importe quelle plateforme et enveloppé dans une couche XML. Il doit pouvoir être découvert et invoqué dynamiquement par d'autres services.

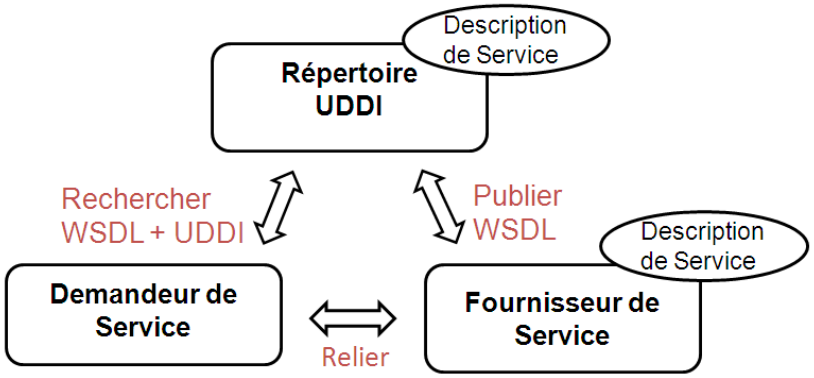



FIG. 2.2 – Architecture orientée-service (SOA)

Cette technologie qui est initiée par IBM et Microsoft, puis en partie normalisée par le W3C, est maintenant acceptée par l'ensemble des acteurs de l'industrie informatique. C'est surtout ce point qui fait des services web une technologie révolutionnaire. L'architecture des services web s'est imposée (tout comme le langage XML) grâce à sa simplicité, à sa lisibilité et à ses fondations normalisées. Le concept des services web s'articule actuellement autour des trois acronymes suivants (Figure 2.2) :

[1] http ://www.cs.wustl.edu/ schmidt/corba-overview.html
[2] http ://www.w3.org/

– SOAP (Simple Object Access Protocol) est un protocole d'échange inter-application indépendant de toute plateforme, basé sur le langage XML. Un appel de service SOAP est un flux ASCII encapsulé dans des balises XML et transporté via le protocole HTTP.

– WSDL (Web Service Description Language) donne la description au format XML des services web, en précisant les méthodes pouvant être invoquées, leurs signatures et le point d'accès (URL, port, etc..). C'est en quelque sorte l'équivalent du langage IDL [3] (Interface Description Language) pour la programmation distribuée CORBA.

– UDDI (Universal Description, Discovery and Integration) normalise une solution d'annuaire distribuée de services web, permettant à la fois la publication et l'exploration. L'UDDI se comporte lui-même comme un service web dont les méthodes sont appelées via le protocole SOAP.

#### 2.2.1.1 Application *Versus* Services web

Selon (Maamar et al., 2003), une application est un service web si elle est : (i) indépendante autant que possible des plateformes spécifiques et des paradigmes de calcul ; (ii) développée principalement pour des situations inter-organisationnelles plutôt que pour des situations intra-organisationnelles ; et (iii) facilement composable (sa composition avec d'autres services web n'exige pas le développement des adaptateurs complexes). Par contre, les services web manquent de mécanismes pour faire face à la qualité de service (QoS) (Menasce, 2002). En effet, les exigences des utilisateurs peuvent avoir lieu sur des aspects fonctionnels et non-fonctionnels, alors que la pile des protocoles des services web soutient la spécification, la publication et la découverte des services en utilisant seulement des aspects fonctionnels. En conséquence, afin de pouvoir utiliser un service web, un système doit avoir la capacité de le tester en se basant sur ses qualités telles la disponibilité, la performance, la sécurité, l'interopérabilité, etc. Ces attributs sont cruciales avant de pouvoir exploiter un service web dans un domaine collaboratif ou non collaboratif.

#### 2.2.1.2 Site web *Versus* Services web

Pour (Wright, 2005), il est important de distinguer les services web (adoptée par le terme "service-oriented architecture") et les sites web : *"Un service web est une composante active dans un environnement qui fournit et gère l'accès à une ressource, qui est essentiel pour le fonctionnement d'autres entités dans l'environnement, alors qu'une page web est statique et fournit une seule représentation de certaines informations"*.

Par contre, certains auteurs (Neward, 2003) argumentent que les services web doivent garder les deux termes du "web" et de "services" qui semblent fondamentales pour la définition du terme : *"Dans le terme service web, il existe deux concepts de base qu'on mélange et qui sont ambigües. L'interopérabilité entre les langages, outils et plateformes, et services, signifie une philosophie de conception visant à corriger les défauts que nous avons découverts avec les objets distribués. Ces deux idées, alors qu'elles sont certainement complémentaires, restent indépendantes, et une étude rapide de chaque terme révèle*

[3] $http://en.wikipedia.org/wiki/Interface-description-language$

*qu'en combinant l'interopérabilité avec les services, nous créons des «choses» qui peuvent efficacement rester autonomes pour l'avenir prévisible".*

#### 2.2.1.3 SOA *Versus* Services web

Une distinction existe entre les termes service web et architecture orientée-services (SOA) (Dutra et al., 2010) (Wright, 2005), qui sont souvent mal distingués dans la littérature. En effet, la SOA est une méthodologie de conception pour l'organisation et la réutilisation des services, vus comme des composants dans un système encore plus large. La distinction majeure entre les deux termes est que la SOA est une architecture, alors que les services web sont une technologie, ou une implémentation d'une architecture logicielle. Ainsi, la SOA se situe à un niveau plus élevé et plus abstrait que les services web, comme une infrastructure qui se focalise sur la façon dont les unités qui composent un système interagissent ensemble à travers une variété d'environnements.

#### 2.2.1.4 Comportement d'un service web

*"Un comportement de service web est défini par l'ensemble des activités que ce service offre, ainsi que la correspondance de ces activités à des échanges de messages."* (Varga and Hajnal, 2003).

Alors que les services web sont des composants avec des interfaces clairement définis, l'architecture logicielle nécessite des processus de vérification et de validation des comportements dues à la composition et à la coordination des services, afin de répondre aux besoins et aux comportements des utilisateurs. L'analyse des comportements d'un service web consiste à analyser principalement deux aspects. Le service web normalement définit son identité et ses interactions par son fichier de description (WSDL). Le comportement d'un service web est formé à partir des opérations basiques d'invocations de ce service, qui constituent les éléments pour l'analyse des comportements d'interaction. Idéalement, la composition de ces services devrait ressembler à l'assemblage des pièces d'un "puzzle" en sachant d'avance qu'ils vont s'emboiter parfaitement dans tous les aspects possibles, mais malheureusement c'est encore loin de la situation actuelle.

### 2.2.2 Orchestration et Chorégraphie

L'orchestration de services web vise à fournir un modèle de conversation (Peltz, 2003a) entre différents services faiblement liés, qui est semblable à une conversation téléphonique, comprenant une série d'échanges entre les parties d'une façon plus souple et dynamique. C'est une description d'un processus de gestion exécutable qui peut interagir avec des services web internes et externes. Une analogie fréquemment utilisée dans l'orchestration est le chef d'orchestre, qui suggère un contrôle centralisé des middlewares traditionaux. La chorégraphie, cependant, est plus descriptive et plus collaboratif dans la nature, dans laquelle chaque partie impliquée dans le processus, décrit le rôle qu'ils jouent dans l'interaction. Ainsi, La différence entre l'orchestration et la chorégraphie de services, est la présence ou l'absence d'un "chef d'orchestre" (Svirskas et al., 2005). L'orchestration implique une délégation (abandon) de la responsabilité au chef d'orchestre qui peut commander les services et la gestion des erreurs, ce qui permet de moins surcharger les services, mais est, par nature, dépendante de ce chef d'orchestre. En revanche, la chorégraphie repose sur chacun

des services afin de répondre de façon appropriée lorsque des erreurs ou des problèmes se produisent, et cela exige de chaque service de comprendre les buts et le contexte global (d'où la sémantique). Il en résulte une charge de responsabilité plus grande sur les services.

En effet, l'orchestration et la chorégraphie sont des efforts qui peuvent donner des solutions à long terme pour les organisations afin de connecter divers services ensemble. Les deux normes les plus répandues dans le domaine c'est le WSCI (Peltz, 2003b), ou le "Web Service Choreography Interface" et le "Business Process Execution Language for Web Services"[4] (BPEL4WS), qui sont conçus pour réduire la complexité inhérente de la connexion entre services web. Nous pouvons voir les protocoles des services web dans la figure 2.3. Ces normes reposent sur des exigences techniques afin d'assurer :

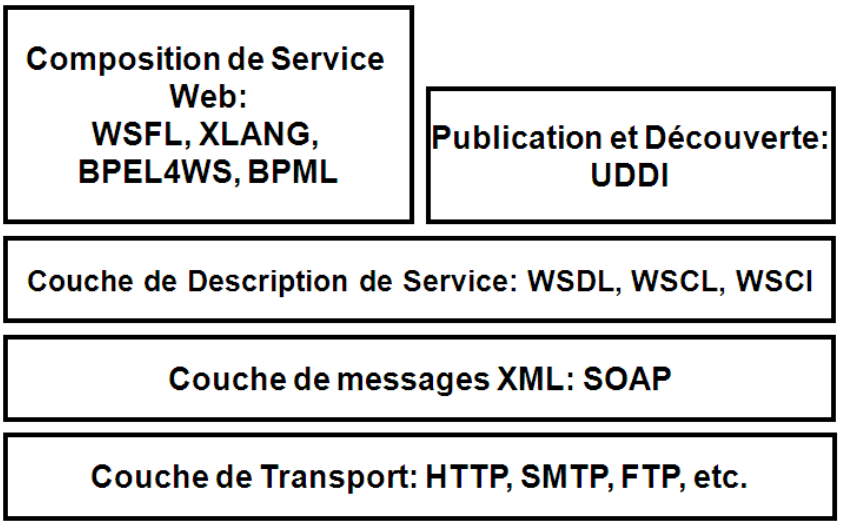



FIG. 2.3 – Représentation en couches des protocoles de services web

- La flexibilité : en prévoyant une articulation claire entre la séparation de la logique du processus et les services web invoqués, qui peut être réalisée par le biais d'un moteur d'orchestration qui gère l'ensemble de flux des processus.
- Activités basiques ou structurées : un langage d'orchestration doit soutenir des activités à la fois pour communiquer avec d'autres services web et de gérer des flux de travail sémantique. Les activités structurées gèrent l'ensemble du processus, en précisant ce que les activités doivent réaliser et dans quel ordre.

### 2.2.3 Services web sémantique

Une limitation partagée par les standards de description basés sur le langage XML, c'est leur déficit d'exprimer la sémantique des informations. Par exemple, deux descriptions identiques en XML peuvent avoir une signification totalement différente, et dépendront du contexte où elles sont utilisées. Par exemple, un demandeur de services ne sait pas quels sont les services fournis à un moment donné. Dans ce cas, il contacte directement le fournisseur pour récupérer la liste des services. En plus, les fournisseurs et les demandeurs possèdent des connaissances différentes du même service proposé. En effet, pour utiliser un service web, un agent logiciel doit avoir une description interprétable et les moyens pour accéder à ce service. Ainsi, une mise en correspondance de services web est

[4]http ://www.ibm.com/developerworks/library/ws-bpelcol1/

nécessaire pour une découverte efficace, et a besoin d'un modèle de données (metadata) bien riche et flexible, qui n'est pas actuellement supporté par l'UDDI.

Le web sémantique est centré autour des métadonnées, des ontologies et des agents logiciels. En effet, les services web sémantique (Nandigam et al., 2005) forment une synergie entre le web sémantique et les services web, et ont le potentiel d'apporter une valeur ajoutée par la découverte autonome, et l'assemblage des services web afin d'accomplir une tâche de domaine. La philosophie et le cadre des services web sémantique est également connu sous le nom de "Service-Oriented Computing (SOC)" (Huhns and Singh, 2005). Nous pouvons voir dans la figure 2.4 la convergence du web traditionnel et des services web vers le web sémantique.

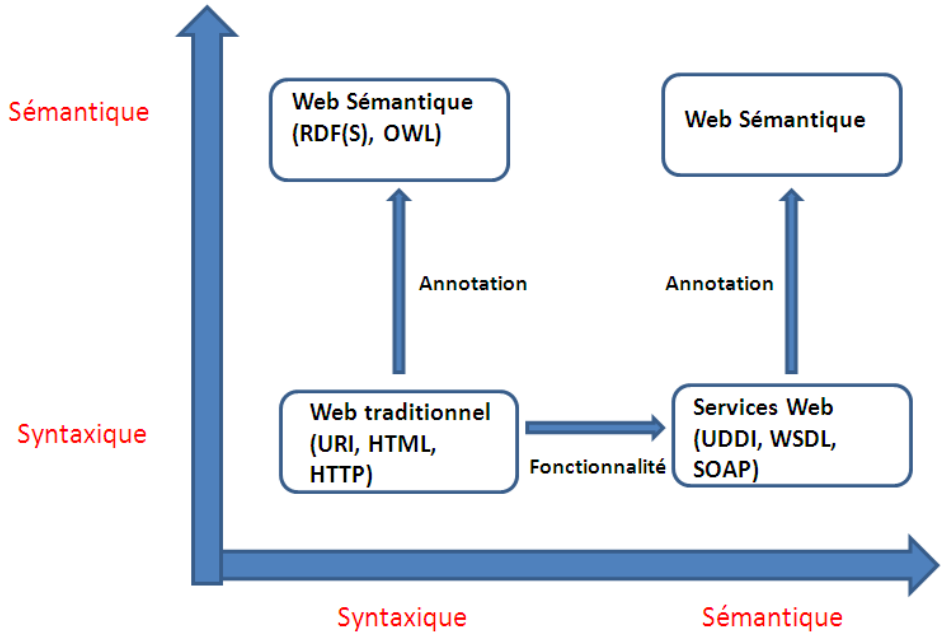



FIG. 2.4 – Service web et le web sémantique

Parmi les ressources web les plus importantes, sont celles qui offrent des "services". Par service, les auteurs (Nandigam et al., 2005) signifient des sites web qui ne se contentent pas seulement de fournir des informations statiques, mais de permettre d'effectuer une certaine action ou un changement dans le monde, comme par exemple la vente d'un produit ou le contrôle d'un périphérique physique. Le web sémantique devrait permettre aux utilisateurs de localiser, sélectionner, utiliser, composer, et de contrôler ces services web automatiquement. Un objectif important est d'établir un cadre dans lequel des descriptions sémantiques sont faites et partagées. C'est pour cela que les sites web devraient être en mesure d'employer une ontologie afin de déclarer et de décrire les services, dites services web sémantique.

#### 2.2.3.1 Ontologie : Définition

Ainsi, les ontologies offrent un remède aux contraintes du web classique. Pour (Blois et al., 2007) une ontologie est définit comme "la science ou l'étude de l'être", qui est un ensemble de termes et de connaissances, y compris le vocabulaire, la sémantique des

interconnexions, et des simples règles d'inférence et de logique pour certains domaines populaires. Par exemple, l'ontologie d'une recette de cuisine comprend les ingrédients, la façon de mélanger et de combiner les ingrédients, les attentes des produits qui seront mangées ou bues, et ainsi de suite. Les ontologies sont utilisées par les personnes, les bases de données et les applications qui doivent partager l'information du domaine (la médecine, la lutte contre le terrorisme, l'imagerie, la réparation automobile, etc.). Une importante réalisation est le développement d'une nouvelle génération de langages web, qui permettent la création d'ontologies de n'importe quel domaine, et leurs instanciations dans la description de sites web spécifiques. Les ontologies sont explicitement mentionnées dans un langage formel, où plusieurs ont été proposés dans la littérature. Cependant, les travaux les plus communs tournent autour du langage RDF [5] et OWL-S [6] :

**RDF ou Resource Description Framework** représente la première étape vers un langage web pour les ontologies. C'est un modèle de données permettant de décrire les ressources sur le web. Plusieurs outils ont été développés pour le RDF, et un nombre considérable d'applications du web sémantique utilisent ce langage, avant qu'un autre langage plus expressif soit développé.

**OWL-S** vise à surmonter les limitations d'expressivité du RDF, en fournissant des moyens de décrire les relations entre les classes (disjonction, union, intersection), la cardinalité et les caractéristiques des propriétés (la transitivité, la symétrie), l'égalité, etc. L'OWL-S fournit des composants pour encoder des connaissances en forme d'ontologie, en utilisant le langage XML. Nous pouvons voir un extrait du langage OWL-S (Figure 2.5) extraite de (Sabou, 2006), qui décrit le fournisseur des équipements médicaux dans le domaine de la santé.

```
<owl:Class rdf:ID="MedicareProvider">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="certifiedBy"/>
      </owl:onProperty>
      <owl:hasValue>
        <HealthProgram rdf:ID="Medicare"/>
      </owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="HealthCareProvider"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="HealthProgram"/>
```

FIG. 2.5 – Le langage OWL-S

Plusieurs propriétés sont attendues du langage OWL-S :

– Découverte automatique tout en respectant certaines contraintes spécifiées par le client.

[5] http ://www.w3.org/RDF/
[6] http ://www.w3.org/Submission/OWL-S/

– Invocation automatique par un programme informatique ou un agent, en utilisant seulement une description déclarative du service, contrairement à l'agent qui a été préprogrammé pour être en mesure d'appeler ce service.
– Composition et interopérabilité automatique des services web pour effectuer des tâches complexes, en se basant sur une description à haut-niveau de la tâche à faire. Avec l'OWL-S, les informations nécessaires pour choisir et composer des services seront encodées aux sites web de ces services. Ainsi, un logiciel peut être écrit pour manipuler ces représentations avec une spécification des objectifs de la tâche.

Une structure d'ontologie de service est motivée par la nécessité de fournir trois types essentiels d'information, comme nous pouvons voir dans la figure 2.6 extraite du site web du W3C[7] :

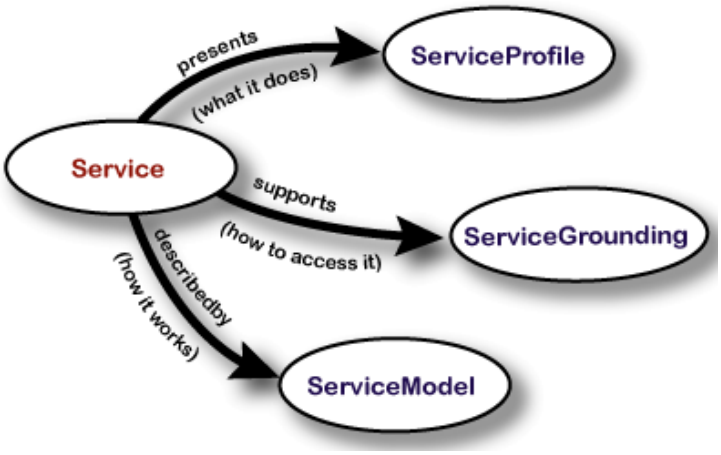



FIG. 2.6 – Ontolgie de service web

– Qu'est-ce que le service fournit pour les futurs clients ? Le "ServiceProfile" répond à cette question, et est utilisé pour publier le service.
– Comment le service est utilisé ? Le "ServiceModel" répond à cette question, où les instances du service utilisent la propriété de "DescribedBy" pour référencer le ServiceModel.
– Comment peut-on dialoguer avec le service ? Le "ServiceGrounding" répond à cette question, qui fournit les détails nécessaires sur les protocoles de transport.

### 2.2.4 Intégration de services web et d'agents

De nombreux chercheurs dans le domaine affirment que les services web seuls sont considérés comme passifs, tandis que les agents peuvent fournir des alertes et des mises à jour lorsque de nouvelles informations deviennent disponibles. Malheureusement, ces technologies ont été à l'origine développées séparément avec différentes normes et caractéristiques. En conséquence, leur intégration devient importante dans ce contexte. Notre idée est d'exploiter les capacités proactives d'interaction des agents afin d'améliorer le comportement des services web dans une architecture collaborative et orientée-service. Avec ce paradigme, les composants logiciels où chacun représente un service et un agent en collaboration, vont interagir pour fournir des services unifiés. Ceci correspond bien avec

[7]http ://www.w3.org/Submission/OWL-S/

la prédiction de (Huhns and Singh, 2005) *"Les agents deviendront une partie essentielle de la plupart des applications web, servant comme une colle qui permet à un système aussi grand que le web d'être maniable et viable"*.

#### 2.2.4.1 Approches pour l'intégration

Plusieurs approches existent dans la littérature afin d'intégrer les services web et les agents dans des divers domaines d'applications :

(Foukarakis et al., 2007) proposent une plateforme qui fournit un temps d'exécution rapide d'un environnement d'agents qui se situe à l'intérieur d'un conteneur web, ce qui ajoute des fonctionnalités d'agents aux serveurs web existants. Les composants de la plateforme sont déployés en tant que services web, avec SOAP (Simple Object Access Protocol) sur HTTP agissant en tant que canal de communication par le biais de messages XML standards. De cette manière, le support d'agent mobile peut être ajouté à l'infrastructure web existante, sans besoin de remplacer les composants ou d'installer un logiciel client.

(Matskin et al., 2005) proposent une solution pour permettre la sélection et la composition des services web avec des agents logiciels appliquées au domaine du marché ("MarketPlace"). L'utilisation des agents permet de soutenir la pro-activité et l'autonomie du processus de composition, par l'intermédiaire des opérations autonomes et de la négociation. L'architecture logicielle proposée permet ainsi une flexibilité et une extensibilité suffisante, tout en offrant un ensemble d'outils intégrés, ainsi qu'un support pour la communication et la coordination.

(Starr et al., 1996) proposent un agent, "Do - I - Care", qui a été conçu pour aider les utilisateurs à découvrir les changements intéressants sur le web. Les agents Do - I - Care automatisent des visites périodiques sur des pages sélectionnées afin de détecter des changements intéressants aux noms des individus. Ils utilisent l'apprentissage automatique pour identifier, par exemple, les changements qui sont intéressants et à quelle fréquence ils se produisent. Une fois qu'un agent détecte un changement, ce dernier est annexé à la page web associée à l'agent, qui est également utilisée pour la pertinence des commentaires et l'activité de la collaboration.

(Maamar et al., 2003) proposent une approche à trois niveaux (intrinsèque, fonctionnel et comportemental) pour intégrer les services web et les agents. Les auteurs affirment que rechercher des services, les intégrer dans un service composite, les déclencher et suivre leurs exécutions sont parmi les opérations que les utilisateurs vont prendre en charge. En même temps, la plupart de ces opérations sont complexes et répétitifs, avec une grande partie adaptée à l'outil informatique et à l'automatisation. Ainsi, les agents logiciels sont des candidats appropriés pour assister les utilisateurs dans leur opérations.

(Buhler and Vidal, 2002) présentent une intégration symétrique des services web et des agents conforme aux standards de la FIPA [8], afin de créer une architecture dans le contexte du projet AgentCities. Les auteurs argumentent que les services web ont été développés sans le concept des agents et peuvent exister sans agents. Ainsi, une approche

[8] http ://fipa.org/

basée "proxy" permet aux deux plateformes d'évoluer en parallèle sans imposer aucune restriction l'une sur l'autre. Cette approche accepte une égalité entre les rôles des agents et des services web, ce qui est différent de la vue traditionnelle que les agents sont considérés sur un niveau plus élevé, et prennent seulement les rôles des fournisseurs et des consommateurs de services web.

(Nguyen and Kowalczyk, 2005) proposent une solution améliorée, WS2JADE, pour l'intégration des services web avec un système multi-agent. Cette approche possède une couche d'interconnexion qui contient des agents spéciaux, appelé "Web Service Agent" ou WSAG. Ces WSAGs collent les agents et les services web ensemble, ainsi, ils offrent des services web en tant que leurs propres services. La deuxième couche s'appelle couche de gestion, qui est capable de faire une découverte active des services et de produire et de déployer automatiquement des WSAGs en temps d'exécution. La combinaison de deux couches statique et dynamique est un outil distinct de WS2JADE par rapport aux différents outils existant, qui est conçu pour découvrir activement des services web et les enregistrer dans le système multi-agents.

#### 2.2.4.2 Comparaison et avantages d'intégration

Les services web sont de plus en plus utilisés pour fournir des comportements actifs sur internet, et promettent aux utilisateurs finaux des avantages qui ont été associés, dans des travaux antérieurs, avec des systèmes multi-agents. Il est donc naturel de considérer quels types de relations existent ou doivent exister entre les agents et les services web. Nous argumentons qu'ils sont des composants distincts, bien qu'ils peuvent partager des buts communs. Dans la littérature, l'idée de composer des systèmes à bases d'agents logiciels a reçu une attention considérable. Plus récemment, des chercheurs ont étudié l'utilisation de services web à bases de RPC (Remote Procedure Call) afin de répondre à des objectifs pour la construction des systèmes flexibles (Dickinson and Wooldridge, 2005). Aujourd'hui, l'interopérabilité suggère que les services web acceptent des documents XML comme entrée, et la restitution d'un autre document XML comme sortie. En effet, l'un des volets de la recherche et de développement sur les agents étend les architectures à bases de messages, afin d'exploiter des propriétés tels que la réactivité, la proactivité, l'autonomie, etc. Afin de comprendre le but de l'intégration de ces deux technologies, nous représentons les avantages et les inconvénients de leurs utilisations :

Les avantages des services web :
- Initiés par des grands acteurs de l'industrie informatique tels IBM et Microsoft.
- Utilisent l'infrastructure déjà déployé pour le web et le HTTP.
- Séparent la spécification de l'implémentation.
- Utilisent des protocoles et standards ouverts et standardisés.
- Permettent une interopérabilité entre fournisseurs hétérogènes de services.
- Permettent une composition facile pour former un service à valeur-ajouté.
- Réutilisent des services dans une infrastructure.

Les inconvénients des services web :
- Complexité de code pour élaborer une requête SOAP.
- Possibilité de saturation port 80 (HTTP) - Disponibilité complète impossible.
- Difficulté de lire les interfaces de descriptions à bases de XML.

– Description syntaxique des interfaces.
– Manque de sémantique - nécessite l'intervention humaine.

Pour les agents, les avantages les plus pertinents sont :
– Simplifier le calcul distribué.
– Mettre en œuvre des systèmes collaboratifs.
– Incorporer des agents en tant que gestionnaires de ressources.
– Assurer une autonomie : ne nécessite pas l'intervention humaine.

Alors que les inconvénients sont ;
– Rarement utilisable dans le monde industriel
– Présentant un risque d'excès d'autonomie qui peut créer des problèmes.

Nous dréssons ci-dessous un tableau comparatif de ces deux technologies.

| Propriétés/Technologies | Services Web | Agents Logiciels | WS + Agents |
|---|---|---|---|
| Usage grand échelle | + | - | + |
| Distribué | + | - | + |
| Proactivité | - | + | + |
| Collaboration | - | + | + |
| Interopérabilité | + | + | + |

TAB. 2.1 – Comparaison entre services web et agents

Cette étude nous montre qu'en combinant ces deux technologies, nous obtenons une synergie d'intégration qui remédie la faiblesse de chaque technologie tout en renforçant leurs avantages individuelles.

### 2.2.5 Bilan

Ils existent plusieurs façons de considérer et d'étudier la relation entre les services web et les agents logiciels (Dickinson and Wooldridge, 2005). Ainsi, nous pouvons distinguer trois façons de concevoir la relation entre ces deux technologies :

– La première vue affirme que les deux technologies ne sont pas conceptuellement distinctes, et donc il n'existe pas une différence de conception entre les deux, qui sont représentés comme des modules fonctionnels actifs dans une architecture faiblement couplée. Les auteurs critiquent cette vue en suggérant qu'ils ont une distinction assez importante à la fois pour le système, les concepteurs et les utilisateurs. Par exemple, si un agent est capable d'agir d'une manière proactive pour atteindre les objectifs d'un utilisateur, il se manifestera dans son interface d'une autre manière que des composants non-agents qui ne possèdent pas ces propriétés. Les auteurs proposent que les agents sont vus en termes de comportements et d'intentions (de l'utilisateur ou de l'agent), qui représentent la différence clé entre les services web et les agents.

– Une deuxième vue est que les agents et les services web peuvent interagir par une initiation de communication par l'un d'eux. En d'autres termes, les agents peuvent invoquer des services web, et vice versa. Des travaux existants montrent qu'il est possible pour les services web d'appeler un agent à condition qu'une correspondance

d'un WSDL à un message ACL[9] (Agent Communication Language) existe. Les auteurs argumentent que ce type d'invocation peut créer certaines complications : l'agent doit exposer des comportements prédéterminés ce qui rend l'invocation de service plus facile à écrire, mais provoque une violation de l'autonomie de l'agent. Dans ce cas, si l'agent invoqué n'est pas fixe et déterministe dans ses comportements, le service web qui l'a invoqué doit se comporter comme un agent lui même, afin de s'adapter à l'autonomie des réponses de l'agent. En conséquence, si le comportement du service web ne se distingue pas d'un agent autonome, il devrait théoriquement être considéré comme un agent, pas un service.

– La troisième vue affirme qu'un élément clé des services web est leurs compositions dans un flux de travail (workflow), pour former des comportements composites complexes. Dans cette optique, les agents sont chargés principalement de la médiation entre les besoins des utilisateurs avec les stratégies et les plans disponibles, où les agents invoquent des services web basiques ou composés selon la nécessité.

Dans notre travail, nous suivons la troisième vision, où les services web sont invoqués par des agents logiciels comme étant des comportements composables. Ainsi, l'autonomie et l'intention se présentent uniquement au niveau des agents logiciels, qui sont représentés comme des composants dans une architecture logicielle collaborative. En effet, le fait d'utiliser les services web est pour changer le comportement du système sans arrêter son exécution, et donc éviter de coder manuellement. Ainsi, une certaine malléabilité est créée, en donnant la possibilité aux utilisateurs d'adapter les services selon la tâche à effectuer.

Dans cette section, nous avons discuté l'importance des systèmes multi-agents dans l'implémentation des systèmes collaboratifs. Ensuite, nous avons présenté les services web, afin de les mettre en œuvre dans le domaine du travail collaboratif. Nous avons discuté les apports que cette technologie peut apporter pour l'implémentation des systèmes interopérables. Dans la section suivante, nous présentons un état de l'art sur les approches et les méthodologies pour la malléabilité des collecticiels.

## 2.3 La malléabilité des collecticiels

L'émergence du travail collaboratif sur internet est une solution à la grande complexité des systèmes, et les difficultés techniques qui pourraient découler de leurs utilisations. En effet, les utilisateurs répartis géographiquement désirent de plus en plus de travailler ensemble sur une tâche commune, alors que l'utilisation des applications rigides et souvent incompatibles conduit à des problèmes d'interopérabilité. Le besoin de systèmes logiciels flexibles est bien traité dans les domaines de recherche de l'ingénierie logicielle (Wulf et al., 2007). En effet, motivé par la nécessité d'être plus efficace dans les développements des logiciels, les travaux ont été menés pour une meilleure réutilisation de code et une augmentation de l'exploitation des architectures logicielles (programmation orientée objet, orientées composants, et plus récemment orientées services). Dans le domaine du TCAO, la motivation pour la flexibilité est différente : le logiciel doit être souple pour s'adapter aux nouvelles situations de travail dans son contexte d'utilisation. Ainsi, les utilisateurs doivent pouvoir adapter leurs applications en fonction de leurs rythmes et

[9]http ://www.fipa.org/repository/aclspecs.html

méthodes de travail. En revanche, les méthodes qui ont été développées pour une évolution de l'ingénierie logicielle, ne considèrent pas le rôle de l'utilisateur final. Ainsi, la recherche sur la malléabilité des collecticiels provient de l'écart entre la conception et l'utilisation des systèmes de collaboration. La malléabilité offre donc aux utilisateurs la possibilité d'adapter l'application en fonction de leurs besoins, et non pas l'inverse.

Dans cette section, nous procédons par définir la malléabilité, qui est identifiée comme une propriété nécessaire dans les environnements du TCAO. Plusieurs auteurs ont essayé d'étudier ce concept afin de concevoir une architecture générique pour les collecticiels ((Stiemerling, 1997b), (Wulf, 1999), (Teege, 2000), (Kahler, 2001), (Slagter, 2004), (Fernandez, 2005), (Torres et al., 2007)). Cependant, la définition exacte de la malléabilité reste toujours un thème de recherche en plein expansion, vu que plusieurs approches peuvent être adoptées pour arriver à la malléabilité, dont la signification et les fonctionnalités varient d'une approche à une autre. Nous tentons de comprendre la façon dont cette propriété a été étudiée et mise en œuvre dans ces différentes approches, afin de concevoir notre propre architecture logicielle malléable pour les collecticiels.

### 2.3.1 Définition

L'étude de l'impact des sciences humaines sur le domaine du TCAO montre que la malléabilité est une propriété fondamentale qui doit être prise en compte dans la création des systèmes collaboratifs. Plusieurs auteurs ont essayé de définir la malléabilité des systèmes collaboratifs, cependant elle reste toujours ambiguë. Par exemple, (Stiemerling et al., 1999) définissent une application malléable comme un système qui peut être adapté proprement aux changements et diversités de besoins. (Biemans and Ter Hofte, 1999) affirment que la malléabilité est la capacité d'un système d'information à permettre à une personne d'ajuster son application face à des préférences personnelles, et des tâches spécifiques à un contexte. (Bourguin, 2004) souligne qu'une application malléable est à la fois utilisable et modifiable par ses propres utilisateurs, et l'activité de redéfinition correspond elle-même à une des facettes de son utilisation. Evidemment, la malléabilité est une propriété cruciale dans les collecticiels, mais la question persistante est comment implémenter la malléabilité, particulièrement pour les utilisateurs finaux qui ne sont pas nécessairement des spécialistes en informatique.

L'imprévisibilité des besoins qui vont émerger au cours du travail collaboratif, fait qu'il est nécessaire que les systèmes logiciels doivent supporter un travail sensible au contexte. Ce travail s'exprime par exemple, par la situation collaborative qui se présente, la spécificité de la tâche à réaliser, et les paramètres dans lesquels la collaboration prend place (distribution géographique, base logicielle et matérielle, connections réseaux). Pour qualifier cette aptitude, les auteurs dans ce domaine emploient le terme de "tailorability" pour signifier la malléabilité : ce terme exprime la flexibilité et l'adaptabilité d'une application face aux nouveaux besoins auxquels elle est soumise. Les auteurs dans (Morch, 1997) (Morch et al., 1998) décomposent la malléabilité en 3 niveaux :

- Le paramétrage ("customisation") : réalisable par une sélection à établir à travers un ensemble d'options de configuration prédéfinies. Par exemple, modifier l'apparence de la présentation d'un objet, ou éditer les valeurs de ses attributs en les sélectionnant d'une liste de valeurs possibles. L'un des problèmes à ce niveau, est

que l'ensemble des options de paramétrage doit être prévu au moment de la conception de l'application, ce qui limite son potentiel.

– L'intégration : consiste à lier des composants prédéfinis à l'intérieur d'une application, ou entre des applications disjointes. L'intégration va au-delà du paramétrage, en permettant aux utilisateurs d'ajouter des fonctionnalités existantes à l'application sans avoir accès au code d'implémentation sous-jacent.

– L'extension : correspond à une modification au niveau de l'implémentation de l'application, par exemple en ajoutant du code à son programme. Cela pose des contraintes fortes sur le langage utilisé pour coder l'application, parce qu'il doit fournir une grande souplesse, et être facilement maîtrisable pour pouvoir être exploité par l'utilisateur final, sans nuire à son efficacité.

Pour (Bourguin, 2000), la malléabilité n'implique pas obligatoirement la possibilité de redéfinir l'application au cours de sa propre exécution. Cependant, il a souvent été identifié que la malléabilité "à chaud" est favorable au support des activités coopératives, où les sujets peuvent adapter leurs collecticiels sans devoir interrompre leurs activités.

### 2.3.2 Objectifs et défis des logiciels malléables

Les systèmes malléables sont différents des systèmes que nous appelons systèmes adaptatifs (Wulf and Golombek, 2001). Alors que les systèmes malléables permettent aux utilisateurs d'avoir le contrôle sur le processus de modifications de l'application, les systèmes adaptatifs construisent ce que nous appelons un modèle d'utilisateur, où ils modifient automatiquement le comportement du système en initiant, proposant et exécutant des modifications possibles. Cependant, un tel système ne peut pas forcément anticiper les préférences des utilisateurs pour qu'il génère les modifications appropriées. C'est pour cette raison que la recherche concernant les systèmes malléables doit forcement augmenter le contrôle sur la modification du système par les utilisateurs finaux. La malléabilité devient l'extension ou la modification d'une application en créant des artefacts persistants, ou bien avoir la possibilité de choisir entre plusieurs comportements anticipés. Même encore, créer un nouveau comportement des composants du système.

La malléabilité est un art humain et technique afin de modifier les fonctionnalités de la technologie en cours d'exécution. La raison principale pour qu'un logiciel soit malléable est la complexité de son utilisation, ainsi que de la tâche à réaliser. Les auteurs (Kahler, 2001) fournissent trois raisons essentiels pour qu'un logiciel soit malléable :

– Les diversités multidimensionnelles que la malléabilité doit prendre en considération, afin de mettre en œuvre un logiciel apte à supporter des utilisations différentes.

– Le dynamise du travail individuel et organisationnel qui correspond à la nature changeante du travail, oblige le logiciel à lui même changer au cours du temps.

– L'incertitude et l'ambigüité dues aux pratiques du travail obligent l'utilisation des méthodes alternatives pour réaliser des tâches.

Des recherches empiriques ont indiqué deux défis majeurs dans la construction des systèmes malléables (Wulf et al., 2007). Le premier défi est de soutenir la re-conception au cours de l'utilisation, et la seconde, de permettre aux utilisateurs de prendre un rôle majeur dans la restructuration de l'infrastructure. En particulier, pour le deuxième défi, d'importantes améliorations ont été décrites, comme le support de la malléabilité pour remédier au problème des compétences divergentes des utilisateurs. Dans ce cas, les utilisateurs avec peu d'expériences, peuvent déléguer une partie des activités de malléabilité complexes aux développeurs.

### 2.3.3 Conception d'applications malléables

En effet, les logiciels de conception suivant le modèle du "Waterfall" (Figure 2.7 extraite de (Laurillau, 2002)) sont préoccupés par la capture, la réalisation et le test d'un ensemble d'exigences. Ceci reflète une vue partielle d'un domaine d'application, avec des exigences particulières en matière de la fonctionnalité et de l'interface. Ce type de modèle suppose que les exigences d'un système sont bien claires au début de la phase de conception, et sont stables pour une longue période. C'est pour cette raison que les approches récentes en génie logiciel essayent de trouver des solutions aux changements dynamiques des exigences. Dans ces approches, les utilisateurs sont activement impliqués dans la procédure de conception. Ils ont la possibilité d'articuler leur besoins selon le domaine d'application, comme nous pouvons voir dans la figure 2.8 extraite de (Stiemerling, 1997b).

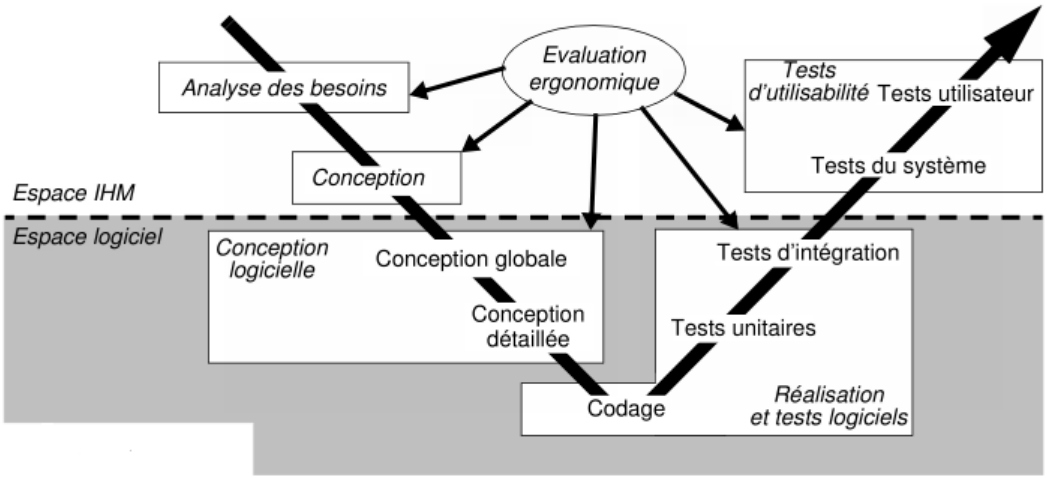



FIG. 2.7 – Modèle de Waterfall extrait de (Laurillau, 2002)

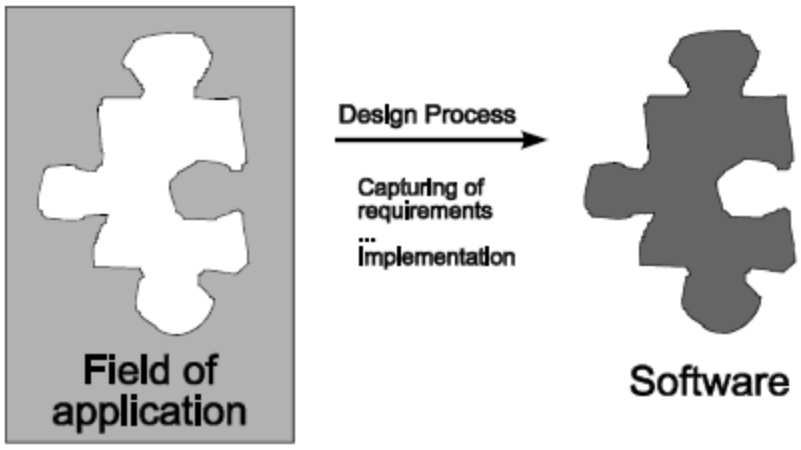



FIG. 2.8 – Illurstration d'une application Malléable extrait de (Stiemerling, 1997b)

Certains auteurs dans le domaine (Stiemerling, 1997b) se posent des questions importantes pour la conception de logiciels malléables ou "flexibles". Ils affirment que pour une bonne conception de ce type d'application, nous pouvons confronter des contraintes telles que :

- Comment le concepteur de l'application peut exprimer les besoins et les exigences éventuelles des utilisateurs, et comment introduire la flexibilité nécessaire dans ces exigences ?
- Comment la flexibilité peut être implémentée techniquement, aboutissant à la question d'architecture logicielle ?
- Comment la flexibilité technique offerte pour une architecture logicielle peut être accessible pour les utilisateurs finaux dans l'interface ?

Ainsi, nous proposons l'utilisation des services web et les agents logiciels, afin de fournir un remède à la question de la "flexibilité technique", qui demeure un souci pour les concepteurs de collecticiels malléables. Pour l'instant, nous faisons un tour d'horizon sur les approches et méthodologies qui visent à intégrer la malléabilité dans les collecticiels.

## 2.4 Approches et méthodologies pour la malléabilité des collecticiels

Il existe plusieurs approches pour introduire la malléabilité dans les systèmes du TCAO, comme les architectures à bases de composants (Stiemerling et al., 1999), (Teege, 1999), (Slagter, 2004) et les architectures réflexives (Bourguin, 2000), qui ont reçu beaucoup d'attention. Cependant, la plupart de ces approches visent des domaines spécifiques, comme le support des collecticiels synchrones, les processus de travail ("workflow") et les tâches collaboratives. Ce n'est pas encore clair si ces approches peuvent être applicables dans d'autres domaines. Dans ce qui suit, nous présentons une étude de ces approches.

### 2.4.1 Théorie de l'activité et la coévolution

(Bourguin, 2000) justifie que la malléabilité possède un fondement théorique permettant de l'appréhender à partir de propriétés fondamentales de l'activité humaine. Il pro-

pose un ensemble de propriétés et d'enjeux dans la construction de collecticiels, dont une propriété fondamentale est la réflexivité. Ainsi, il propose une architecture cadre fondée sur les concepts et les mécanismes proposés par la théorie de l'activité. Les leçons tirées sont mises à profit dans la plateforme CooLDA (Bourguin, 2003), un environnement réflexif du TCAO destiné à mieux comprendre le principe de la coévolution, qui est une approche plus globale et coopérative de la malléabilité.

Ainsi, l'auteur a identifié deux propriétés fondamentales de l'activité humaine :

- La réflexivité qui permet aux concepteurs de l'application d'accéder et modifier sa structure au cours de sa propre exécution.
- La deuxième est la cristallisation ou la réutilisation de l'expérience identifiée, qui dit qu'un artefact transformé au cours des activités par ses utilisateurs cristallise et sauvegarde leurs expériences.

Nous pouvons voir dans la figure 2.9 une capture d'écran de l'environnement DARE (Bourguin, 2004) qui est à la base de la plateforme CooLDA. Ainsi, l'auteur présente une définition de collecticiel par rapport à la théorie de l'activité :

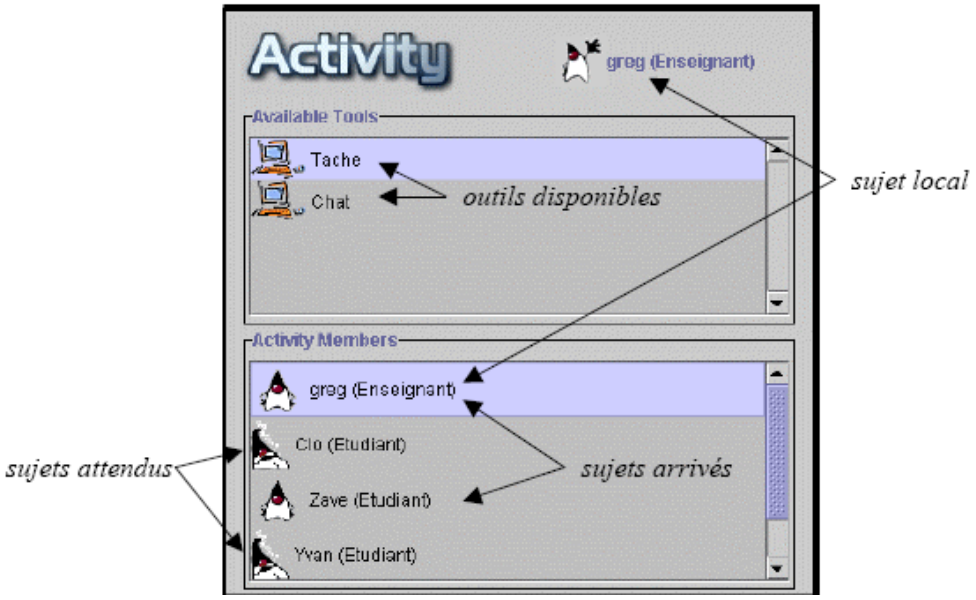



FIG. 2.9 – Capture d'écran de DARE, extrait de (Bourguin, 2004)

- Le collecticiel n'est pas l'activité, mais est un support à l'activité.
- Le collecticiel est un élément médiateur à part entière de l'activité.
- Le collecticiel supporte sa propre activité coopérative de redéfinition.
- Le collecticiel met en œuvre des patrons d'activité réutilisables et adaptables in situ.

En effet, la théorie de l'activité montre qu'un système de TCAO doit être assez malléable pour pouvoir être totalement redéfini au cours de sa propre exécution. Les seuls invariants sont la structure de bases de l'activité, c'est-à-dire son modèle le plus générique, ainsi que les mécanismes de base qui le font vivre.

### 2.4.2 Approche à base de workflow

Les auteurs (Alda et al., 2007) affirment que la capacité des utilisateurs à adapter des flux de travail personnalisés sera de plus en plus importants dans les applications futures. Pour (Dangelmaier et al., 2002), la malléabilité traite l'amélioration de l'utilisation des données venant des processus du travail collaboratif, afin d'optimiser la performance des tâches. Ainsi, ils présentent une approche (Figure 2.10) pour optimiser la malléabilité pour les utilisateurs finaux dans des zones de travail hétérogènes et partagées. Ils démontrent leurs concepts à l'aide des systèmes de gestion de workflow. En effet, les auteurs affirment que dans les systèmes à grande échelle, le problème central lié à la malléabilité surgit de la complexité incrémentale de spécifier une grande quantité de paramètres liés au système et au domaine. Ainsi, la malléabilité traite souvent des propriétés détaillées d'une tâche fortement reliée à des aspects techniques. Elle est souvent une affaire d'experts, qui surgit des différents modèles de travail.

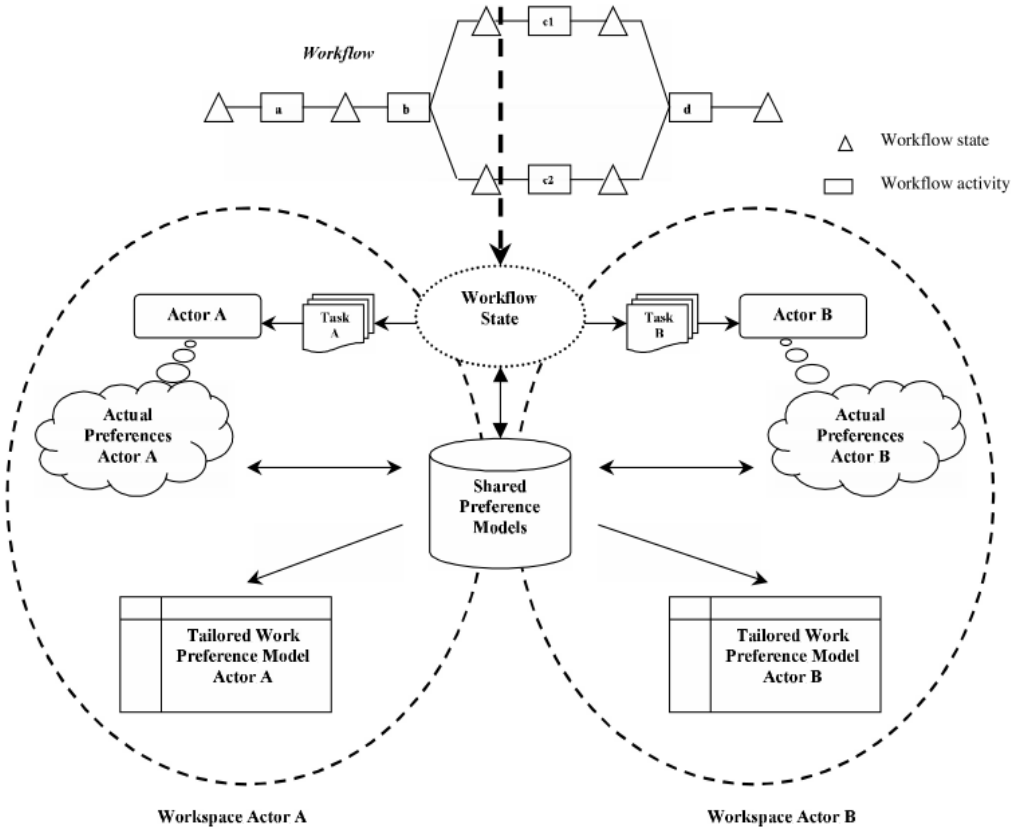



FIG. 2.10 – Illustration de l'approche de la malléablité dans les systèmes de workflow, extrait de (Dangelmaier et al., 2002)

Les questions de collaboration dans les systèmes de workflow sont d'extrêmes importances, où les tâches exigent une connaissance importante des individus. En effet, le support des workflows est principalement défini par la présentation et le rassemblement des données appropriées à travers des interfaces, qui doivent fournir aux utilisateurs une fonctionnalité pour rassembler des informations et partager leurs connaissances. Rendre ces interfaces malléables représente une partie indispensable de cette fonctionnalité. Dans ce contexte, la tâche malléable peut être simplifiée en sélectionnant un modèle de préférence approprié, pour exécuter un travail basé sur la connaissance collective des

parties malléables du système. Un modèle de préférence est présenté en termes de : "Situation" qui spécifie l'environnement préférable pour un modèle de préférence donné, "Decision" qui traite des questions d'implémentation, et "Evaluation" pour communiquer la position des individus.

### 2.4.3 Approche à base d'objets médiateurs

(Syri, 1997) propose une plateforme qui permet l'intégration de services basiques pour le support de la coopération par un développeur. Des parties de cette fonctionnalité sont rendues visibles aux utilisateurs, leur permettant de prendre conscience de la forme choisie de la coopération dans le système. L'utilisateur a donc la possibilité de faire quelques réglages en rajoutant des services supplémentaires, qui ont été développés dans le contexte de la coopération. Dans leur modèle, les services sont appelés "CSCW enablers", qui encapsulent des fonctionnalités de coopération, et les fournissent aux objets dans le système pour les rendre "conscient de la coopération". Tous les "enablers" qui appartiennent à un objet sont administrés par un "CSCW mediator", qui encapsule leurs interactions.

Ainsi, le modèle permet la spécification de la fonctionnalité qui sert comme un support de la coopération, pour qu'elle soit appliquée quand le contexte où les objets sont modifiés. Les médiateurs sont la caractéristique clé qui permet la malléabilité de la coopération au cours de l'exécution. Dans la figure 2.11, nous pouvons voir comment un appel de méthode sur un objet cible est transmis à son médiateur, qui choisit les "enablers" appartenant à l'objet cible, en invoquant les fonctions de coopération offerts par celui ci.

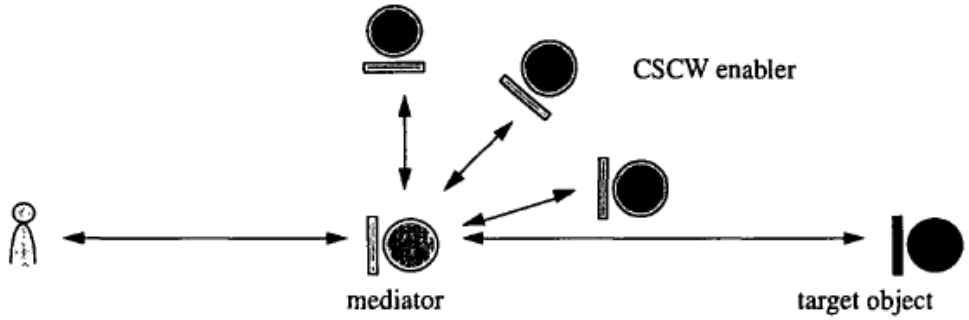



FIG. 2.11 – Approche à bases de médiateurs, extrait de (Syri, 1997)

### 2.4.4 Architectures à base de composants

Beaucoup de recherches ont été réalisées pour la conception des architectures à base de composants. Nous présentons brièvement quelques unes :

- **Approche à base de la réflexivité :** (Stiemerling and Cremers, 1998) proposent le concept d'une application computationnelle réflexive, qui possède une représentation des aspects d'elle-même. Cette autoreprésentation doit être changeable et connectée aux aspects représentés : si la représentation change, l'application change aussi.

- **Approche à base de champs de validité :** Les auteurs (Wulf, 1999) se sont concentrés sur la définition des différents comportements des systèmes par des

champs de validité spécifiques. Ils développent une approche basée sur la malléabilité déclarative, qui montre comment des incohérences résultantes des déclarations contradictoires, peuvent être traitées soit automatiquement ou bien en faisant participer les utilisateurs affectés.

- **Approche à base de modules fonctionnels :** (Slagter et al., 2001) introduisent le concept de la "malléabilité à l'extrême", qui se traduit par l'extension des ensembles des fonctions dans le système avec de nouveaux modules. Un exemple est de permettre à un utilisateur de télécharger des modules de l'internet et les brancher directement dans son système (plugins, widgets etc.). Ainsi, des standards pour l'interopérabilité entre les modules issus des différents fournisseurs sont désormais essentiels. Ils doivent décrire les types des modules qui existent, leurs façons de communiquer et les services qu'ils fournissent.

- **Approche à base d'enrichissement de messages :** (Payet, 2003) propose d'exploiter le concept d'enrichissement des messages dans les composants, qui donne à chaque acteur la possibilité de rattacher des informations à tous les messages qui circulent dans le système.

- **Approche orientée-aspect (AOP) :** Pour (Torres et al., 2007), un collecticiel doit prendre en considération une multitude de préoccupations, qui sont définies par les intérêts qui se rapportent au développement du système, où tout autre aspect critique qui a une ou plusieurs parties prenantes. Ils affirment que la séparation des préoccupations favorise la malléabilité.

En effet, le concept d'un logiciel à base de composants est indépendant de n'importe quel domaine d'application. Ainsi, ces composants peuvent être appliqués dans presque tous les systèmes. En plus, dans ces types d'architectures, la fonctionnalité du système et la fonctionnalité de la malléabilité sont traités comme des problèmes séparés. Nous pouvons donc prévoir qu'une approche mettant en œuvre la malléabilité des systèmes basée sur le concept des composants logiciels, est largement applicable (Roseman and Greenberg, 1997), (Slagter et al., 2000), (Ter Hofte, 1998). Dans ce qui suit, nous présentons quelques modèles d'architecture à bases de composants qui se trouvent dans la littérature, et qui essayent d'intégrer le concept de la malléabilité dans la conception des collecticiels.

### 2.4.5 Quelques modèles de collecticiels malléables

La plupart des approches dans la littérature se reposent sur des architectures à bases de composants pour concevoir des applications malléables. Un moyen possible de créer des collecticiels malléable est de les concevoir à partir de différents modules qui offrent un ensemble de comportements. Dans ce cas, le comportement d'une application est déterminé par l'ensemble de modules qu'ils la composent, d'où un collecticiel est nommé "composable". Dans cette section, nous citons quelques modèles de collecticiels malléables existants dans la littérature.

#### 2.4.5.1 Le modèle Coops

Le modèle Coops (Cooperative People and Systems) (Slagter et al., 2001) est un modèle de collecticiel qui décrit les types de modules fonctionnels qui forment l'application collaborative, ainsi que les relations entre ces modules. Les auteurs décrivent le système comme étant un modèle de référence de services, qui définit les types des modules ou GSMs (Groupware Service Module), qui sont eux mêmes définis en groupant des modules plus basiques, GSME (Groupware Service Module Element), qui forment l'unité de composition des services offert par un collecticiel. Dans ce contexte, un modèle de référence est définit comme étant une structure ou une organisation d'entités fonctionnelles, qui sont reliées et qui définissent des fonctions primaires à un niveau d'abstraction élevé. Nous pouvons voir dans la figure 2.12 la relation qui existe entre les GSMs d'un système collaboratif.

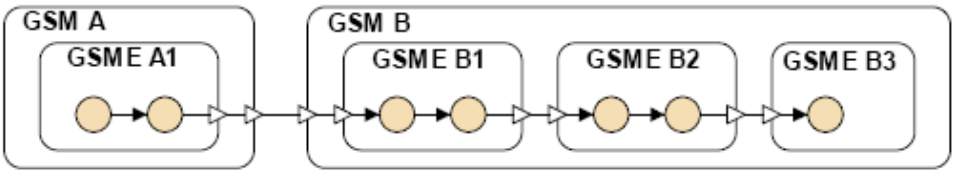



FIG. 2.12 – GSMs et GSMEs constituants le modèle Coops, extrait de (Slagter et al., 2001)

Ainsi, les utilisateurs peuvent sélectionner et composer des GSMs via la sélection des GSMEs, pour introduire une certaine malléabilité sur le comportement de leurs applications collaboratives. Le regrouprement des GSMEs est possible à condition que les critères ci-dessous soient respectés :

- Flexibilité appropriée, où les GSMs doivent être suffisamment larges pour permettre aux utilisateurs de sélectionner des comportements qui répondent à leurs besoins au cours de la collaboration. En même temps, le nouveau comportement ne doit pas aboutir à une surcharge d'informations.

- Les entités de GSMs sélectionnées doivent représenter des groupes de GSMEs pour que les utilisateurs puissent les manipuler. Les auteurs supposent que si les types des GSMs sont conformes avec les types d'unités que les utilisateurs peuvent distinguer, la chance de succès de la malléabilité devient plus grande.

- Inclusion et exclusion de clusters de comportements, où un cluster de GSMEs forme un autre cluster pour un autre type de GSMs. Dans ce cas, les utilisateurs peuvent choisir d'inclure un type de coordination (par exemple un rôle d'accès) dans une conférence, et inclure un différent type de coordination (workflow) dans une autre conférence.

#### 2.4.5.2 La plateforme CoCoWare

La plateforme CoCoWare (Collaborative Component Software) (Slagter et al., 2002) est une plateforme "open-source", qui est une implémentation indépendante de l'architecture proposée dans Coops (section précédente). L'objectif principal de la plateforme est d'aider les concepteurs et les développeurs des applications collaboratives, à créer des

systèmes qui répondent le mieux aux besoins spécifiques d'un cadre de coopération. Ces besoins sont déterminés par les exigences futurs des utilisateurs, les tâches qu'ils effectuent ensemble, et le contexte dans lequel ces tâches sont exécutées. La plateforme est destinée à la création de systèmes collaboratifs qui peuvent être modifiés en cas de besoins. Ceci est effectué suite à un changement dans le groupe d'utilisateurs en coopération, ou un changement de contexte dans lequel les tâches sont exécutées. La figure 2.13 montre les systèmes collaboratifs de deux utilisateurs participants à une conférence, y compris les relations entre les systèmes et celles des composants :

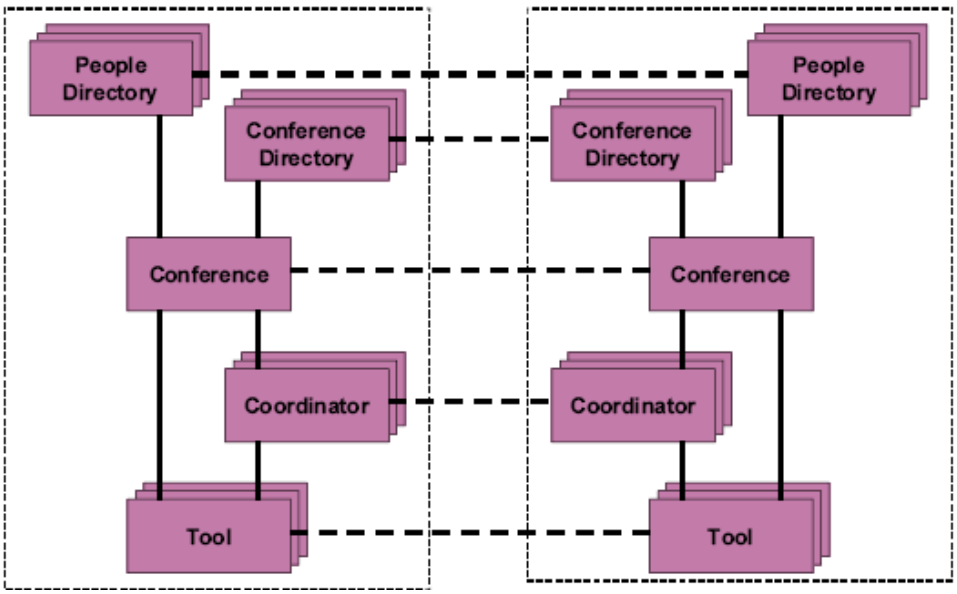



FIG. 2.13 – La plateforme CoCoware.Net, extrait de (Slagter et al., 2002)

– Composant de conférence, qui permet à l'utilisateur de démarrer, arrêter, adhérer ou laisser des conférences. Il permet aussi à l'utilisateur de gérer l'ensemble d'outils utilisé (audioconférence, droit de vote et partage d'applications).

– Composant d'outils, qui reçoit des informations du composant de conférence sur les participants, ainsi que la façon de créer une connexion avec eux (en fournissant une adresse IP, numéro de téléphone, numéro de port, et les types de codage à utiliser).

– Composant d'utilisateurs, qui fournit des informations sur la disponibilité des utilisateurs et des conférences, ainsi que de déclencher activement de nouvelles conférences, et des demandes d'adhésion à des conférences en cours.

– Composant de coordination, qui affecte des rôles à chaque utilisateur afin de mettre en œuvre une politique dynamique de coordination. Cette politique détermine ce que les utilisateurs sont autorisés à effectuer comme type d'actions (lire, écrire ou supprimer les actions sur un outil spécifique).

#### 2.4.5.3 Plateforme FreEvolve et Modèle FlexiBeans

Le modèle FlexiBeans (Won et al., 2006) est une adaptation du modèle JavaBeans de Java, avec l'appui sur les primitives d'interaction des variables partagées (implémentées

comme étant des objets partagés). Les composants atomiques de FlexiBeans sont mis en œuvre et stockés sur le serveur FreEvolve, et regroupés avec d'autres ressources comme les fichiers JAR. Chaque composant peut être instancié dans des compositions différentes en même temps, alors que chaque instance possède son propre état. Afin de permettre la malléabilité au moment de l'exécution, les éléments et leurs représentations abstraites doivent être visualisés sur l'interface de l'utilisateur.

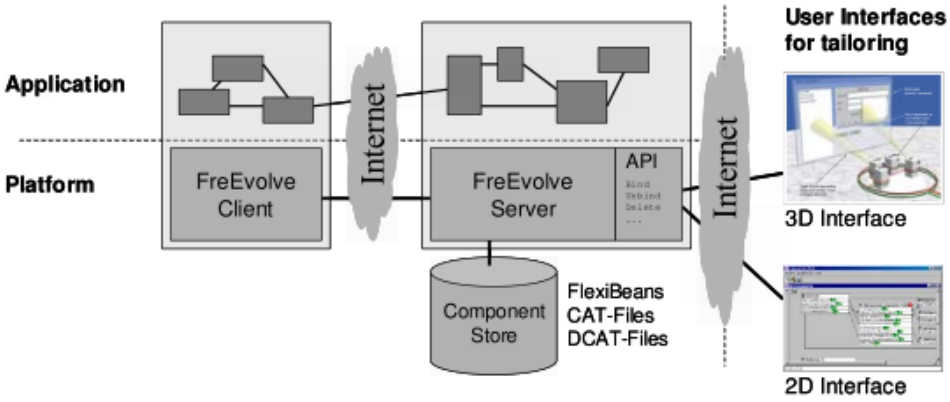



FIG. 2.14 – La plateforme FreEvolve extrait de (Won et al., 2006)

A partir du modèle FlexiBeans qui fournit un moyen d'implémenter les composants (objets) d'une façon standardisé, la plateforme FreEvolve permet le déploiement et la recomposition dynamique de ce modèle. Ce processus est effectué en concevant un langage pour décrire la composition des objets basiques, qui sont visualisés par l'utilisateur comme étant des boîtes noires. Pour cela, les auteurs ont décrit un langage, CAT (Component Architecture for Tailoring) (Stiemerling, 1997a), afin de décrire la composition de ces objets pour constituer des structures plus complexes. Nous pouvons voir l'architecture de la plateforme FreeEvolve dans la figure 2.14.

#### 2.4.5.4 Politeam

Le but du projet POLITeam (Klöckner et al., 1995) est de fournir un support électronique pour le travail coopératif du gouvernement allemand distribué entre les villes de Bonn et Berlin. Le système fournit une coopération asynchrone pour la circulation et le partage des répertoires de travail dans un environnement virtuel. Il utilise une approche participative et cyclique, avec des évaluations par divers ateliers et de visites de lieux de travail. Un exemple de cette d'application est un outil de recherche, où les utilisateurs ont exprimé leurs opinions sur la manière dont l'outil est censé présenter les résultats, et la façon dont la confidentialité des documents des propriétaires peuvent être protégés, par la limitation de l'espace de recherche de l'outil. Une autre question importante est le traitement des résultats de recherche, où certains utilisateurs préfèrent un lien pour accéder aux documents, tandis que d'autres voulaient créer une copie de celui-ci. Il est devenu évident qu'il n'y avait pas une meilleure solution à ces problèmes, d'où, pour les auteurs, l'outil de recherche constitue un excellent exemple pour appliquer des concepts de la malléabilité.

Dans cet outil de recherche "malléable" illustré dans la figure 2.15, les résultats sont intégrés dans un commutateur afin de canaliser l'ensemble des documents trouvés sur les bureaux virtuels des utilisateurs, dans une liste de résultats sur la fenêtre supérieure. D'autre part, tous les autres documents trouvés sont affichés dans la liste de résultats

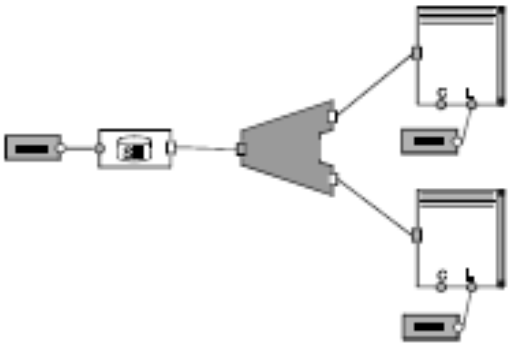


FIG. 2.15 – Un simple example de l'outil de recherche, extrait de (Klöckner et al., 1995)

sur la partie inférieure. L'utilisateur pourra ainsi créer un lien vers un document trouvé, ainsi que vers le composant qui gère la liste de résultats qui offre un port de contrôle permettant de créer une copie de celle ci.

### 2.4.6 Bilan

Nous avons exploré les approches et méthodologies utilisés afin d'introduire la malléabilité dans les collecticiels. Nous pouvons voir dans la figure 2.16 la classification de ces approches. Nous remarquons que les auteurs de ces travaux affirment tous l'importance de la malléabilité afin de s'adapter aux besoins dynamiques et évolutives des domaines d'application. A partir de cette étude, nous déduisons quelques remarques concernant la malléabilité des collecticiels :

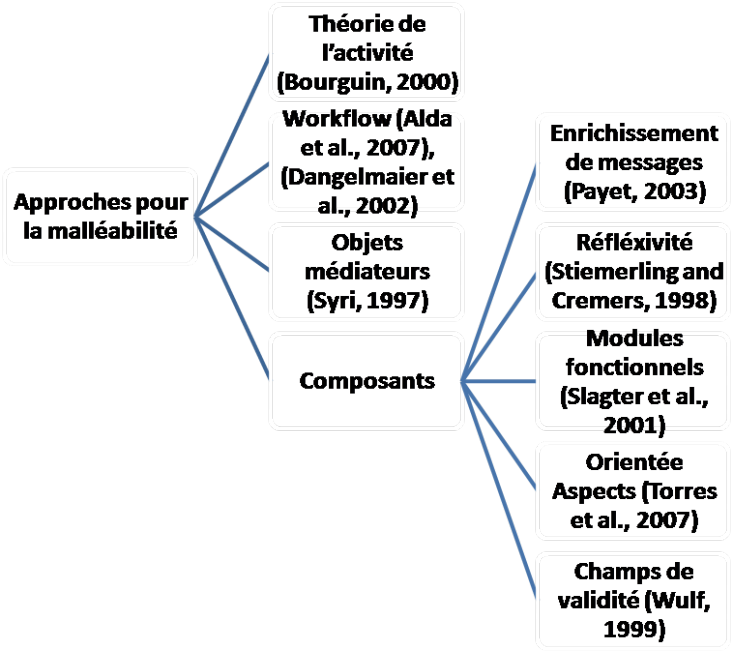



FIG. 2.16 – Classification des approches de la malléabilité des collecticiels

– La malléabilité est une propriété importante qui doit être prise en compte lors de la conception des systèmes de TCAO.

– La plupart des approches dans la littérature s'appuient sur des architectures à bases de composants pour implémenter la malléabilité dans leurs systèmes.

– Beaucoup d'approches existent sans pour autant fournir une approche pour implémenter la malléabilité dans les systèmes de TCAO d'une façon explicite. Ainsi, de nouvelles approches doivent avoir lieu dans le contexte académique en tant qu'une plateforme de simulation, ainsi que dans l'industrie, afin de donner aux utilisateurs des moyens réels d'adapter leurs systèmes à leurs besoins.

## 2.5 Conclusion

Dans ce chapitre, nous avons présenté dans la première section les agents logiciels et les services web. Nous avons étudié la synergie d'intégration de ces deux technologies. Ensuite, nous avons présenté un état de l'art sur les approches pour la conception des collecticiels malléables. En fait, les recherches sur la malléabilité ont commencé suite à l'écart qui existe entre la conception et l'utilisation d'un système. Il existe deux approches qui visent à combler cet écart : une conception centrée-utilisateur et la malléabilité des systèmes. Le résultat principal de nos recherches est le manque d'une plateforme qui introduit explicitement la malléabilité par la création de nouveaux composants interopérables, non prévus par le système lors de sa conception. En plus, les recherches sur la malléabilité ne traitent pas la rentabilité des systèmes malléables, qui est un but essentiel dans la conception des applications collaboratives. Nous avons identifié quelques pistes de recherches qui seront intéressantes pour notre étude sur la malléabilité des systèmes collaboratifs :

– Comment développer une plateforme collaborative malléable et générique ?
– Quelles sont les aspects qui contribuent à la malléabilité des systèmes collaboratifs ?
– Quel modèle d'architecture logiciel faut adopter pour faciliter l'implémentation de la malléabilité ?
– Quel type de malléabilité faut-il pour une performance raisonnable du système collaboratif ?

Ces questions vont nous guider dans nos recherches afin de comprendre le concept de la malléabilité des collecticiels. Le but est d'en déduire une conception d'une architecture collaborative qui met en œuvre ce concept. Cette plateforme doit fournir des perspectives sur des questions comme : pourquoi les utilisateurs veulent la malléabilité ? Quand ils la veulent et comment ils veulent rendre malléables tels systèmes ? Quand ces aspects sont établis, des méthodes d'évaluation et des directives peuvent être développées, et donc des observations de travaux "réels" peuvent commencer.

# Deuxième partie

# CONTRIBUTION

# Chapitre 3

# MODELES D'ARCHITECTURE LOGICIELLE POUR LA MALLEABILITE DES COLLECTICIELS

Dans un monde gouverné par des entreprises qui offrent des services hétérogènes, les organisations essayent d'effectuer collectivement des travaux en mettant en œuvre des workflows communs. Ce processus permet de transmettre des documents, de l'information ou des tâches d'un participant à l'autre d'une manière gouvernée par des règles ou des procédures. D'un point de vue workflow, un logiciel composite peut être considéré comme une suite de services opérée par des données qui doivent être interopérables, où l'interopérabilité est caractérisée par la capacité de deux ou plusieurs composants logiciels à coopérer malgré la différence de langages de programmation, de l'interface, et de la plateforme d'exécution. Les services web constituent une nouvelle génération de services logiciels interopérables et en pleine croissance, où leurs utilisations est en train d'émerger de plus en plus.

Notre chapitre est divisé en deux grandes parties : Dans la première section, nous proposons un formalisme de collaboration machine-machine à base de services web. Nous nous sommes inspiré du modèle 3C afin de le mettre en œuvre dans la collaboration entre les machines connectés sur internet, afin d'échanger des services web. Ensuite, nous décrivons une architecture logicielle, qui se base sur le formalisme présenté. Dans le deuxième section, nous nous basons sur le travail de (Khezami, 2005) (formalisme de SMA-C) pour introduire un formalisme qui intègre les services web et les agents logiciels pour une collaboration malléable. Finalement, nous décrivons notre architecture logicielle malléable pour les collecticiels, $U3D$, qui se basent sur l'intégration des services web avec les systèmes multi-agents. Ainsi, dans ce chapitre, nous présentons un cadre pour la collaboration machine-machine, où la machine est au centre de la collaboration malléable. Nous distinguons ce type de collaboration (que nous appelons collaboration interne) de la collaboration homme- homme (ou collaboration externe).

## 3.1 Quelques définitions et formalismes utiles

A partir de l'état de l'art présenté dans les deux premiers chapitres, nous tentons d'utiliser les concepts et les technologies cités, afin d'introduire une architecture logicielle pour les collecticiels malléables. Ainsi, la malléabilité est mise en œuvre dans notre système par la possibilité d'intégrer ou de composer de nouveaux services, qui n'étaient pas prévus dans la phase de conception. Nous présentons quelques définitions qui nous aident à éclaircir la notion de la malléabilité, ainsi que la notion de service, qui est la composante principale apte à subir les mécanismes de malléabilité.

### 3.1.1 Nouvelle définition de la malléabilité des collecticiels

Comme il n'existe pas encore de définition claire et unanime sur la malléabilité des collecticiels, nous proposons notre propre définition qui couvre les principales exigences sur la malléabilité qui ressort de l'état de l'art.

**Un collecticiel malléable :** Un collecticiel malléable est un collecticiel qui possède une capacité à composer ou à intégrer un ou plusieurs services, sans arrêter l'exécution de l'application.

Ainsi, nous définissons la composition et l'intégration comme suit :

**La composition de services :** La composition est un processus permettant de composer un ou plusieurs services, afin de créer un nouveau service qui possède un nouveau comportement.

**L'intégration de services :** L'intégration est un processus permettant d'ajouter un ou plusieurs services dans l'application.

**Service :** Un service est définit comme étant un composant logiciel qui présente une ou plusieurs fonctionnalités. Un service peut être :

- un service interne, ça veut dire implémenté localement et intégré dans l'application lors de la phase sa conception.

- un service externe, dans ce cas, il est vu comme étant un service web, ou un composant logiciel qui agit d'une façon interopérable dans la collaboration machine-machine, et qui utilisent des protocoles standardisés (section 2.2.1).

En se basant sur le trèfle fonctionnel des collecticiels (3C), nous pouvons ainsi introduire les deux propriétés de la malléabilité qui nous avons retenues (intégration et composition) dans la spécification fonctionnelle des collecticiels, comme illustré dans la figure 3.1 :

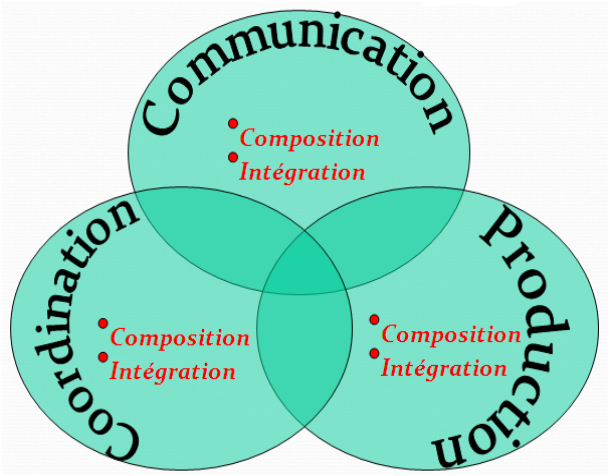



FIG. 3.1 – Prise en compte des deux propriétés de la malléabilité dans le trèfle fonctionnelle des collecticiels

### 3.1.2 Un formalisme de services web pour la découverte et la composition

Il existe plusieurs approches dans la littérature qui traitent de la découverte et de la composition de services web (Akkiraju et al., 2003), (Lamparter and Ankolekar, 2006). Dans (Pathak et al., 2005), les auteurs décrivent une plateforme basée sur une ontologie pour une découverte flexible des services web sémantiques. Cette approche repose sur une correspondance de l'ontologie d'utilisateur avec une ontologie de domaine utilisée pour spécifier des services web sur internet. Les auteurs montrent la façon dont une requête d'un utilisateur pour récupérer un service web qui répond à certains critères de sélection, peut être traitée par un moteur de mise en correspondance, qui est conscient de l'ontologie du domaine et les services web associés. En plus, l'approche montre également comment les préférences des services web spécifiés en termes de besoins non fonctionnels (par exemple la qualité de service ou la QoS), peuvent être incorporées dans le mécanisme de découverte pour générer une liste partiellement ordonnée.

(Lee and Lee, 2007) introduit un modèle formel pour la composition des services web à bases d'agents, qui se reposent sur des descriptions sémantiques ainsi que la QoS. Dans ce modèle, il définit un service web comme suit :

Un service web (WS) est définit par ses fonctionnalités, ses propriétés de QoS, et des informations sur sa localisation.

$$WS =< IOPE, QoS, Loc > \tag{3.1}$$

où

$$IOPE =< input, output, precondition, effect > \tag{3.2}$$

IOPE est composé d'entrée, de sortie, de condition préalable, et les effets d'utilisation du service web.

$$QoS =< q1, q2, ..., qm > \tag{3.3}$$

*qk* est une propriété de qualité de service, telles que la fiabilité, la disponibilité, ou le temps de réponse.
Loc est la localité.

En effet, les IOPEs sont une caractérisation abstraite de ce qu'un service peut faire. Ces propriétés se fondent sur le type de contenu dans l'UDDI, en décrivant les propriétés d'un service nécessaires pour la découverte et la composition automatique. Les IOPEs sont spécifiés par l'ontologie décrite par le langage OWL-S.

### 3.1.3 Un formalisme Multi-Agent pour la collaboration

Les agents logiciels font l'objet de nombreuses recherches dans des divers domaines. Ils ont la capacité de prendre des décisions de façon autonome, sans une intervention humaine et sans influence d'autres agents. Les auteurs dans (Palathingal and Chandra, 2004) soulignent que les architectures à base d'agents fournissent plusieurs avantages sur les technologies orientées-objets, où les objets communiquent par le biais de messages. Dans ce cas, l'expéditeur doit connaître l'adresse du récepteur. En revanche, les agents sont conformes à un protocole de communication (FIPA ACL, etc.) qui permettent à un agent d'envoyer un message à un autre agent sans besoin de connaître son adresse ou bien les méthodes spécifiques à la disposition de cet agent. Ainsi, les agents communiquent entre eux par le biais des messages interopérés afin de se regrouper pour accomplir une tâche. Par exemple, une tâche de communication nécessite l'intervention des agents de communication. Bien évidemment, certaines tâches nécessitent l'intervention des agents de deux espaces différents, voir des agents des trois espaces. Par exemple, afin de créer un service composite de plusieurs services de bases, ca nécessitera donc l'intervention des agents de chaque espace du modèle 3C.

#### 3.1.3.1 Formalisme C4 - Terminologies

Nous proposons une suite des travaux de (Khezami, 2005), qui a proposé le modèle C4 (agent collaborateur) pour la collaboration. Un modèle de collaboration est mis en place, qui est spécifique pour les agents du système dans lequel les interactions internes (Figure 3.2) sont distinguées, ainsi que les interactions externes (les interactions entre les agents collaborateurs). Ce modèle tient compte des propriétés propres aux systèmes multi-agents et des caractéristiques de la collaboration. Les inconvénients de ce modèle (présenté dans la section 1.2.9) nous ont poussés à l'améliorer. Ainsi, dans la nouvelle architecture, l'agent de collaboration est un agent hybride, capable de rechercher de nouvelles ressources sur internet en tant que services web, et les exploiter afin d'offrir aux agents internes de nouveaux comportements. Ce processus améliore la collaboration pour la rendre "malléable".

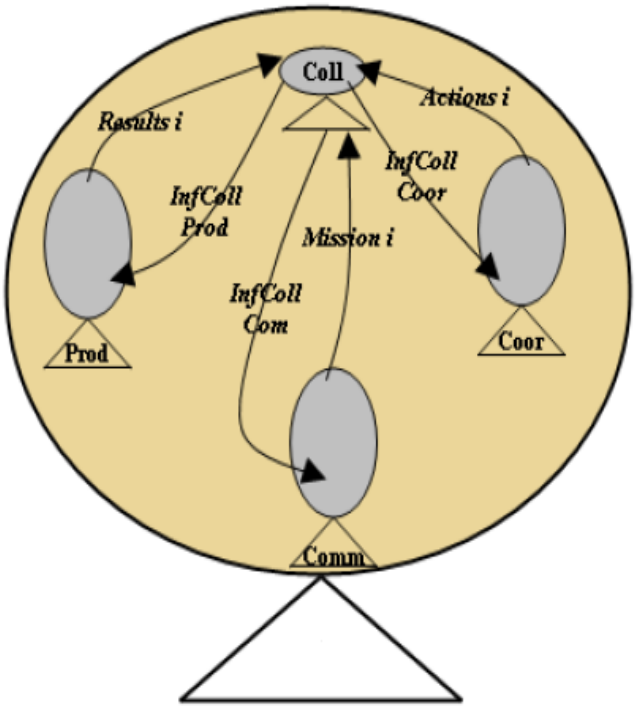



FIG. 3.2 – Interactions internes dans un agent collaborateur, extrait de (Khezami, 2005)

Pour (Khezami et al., 2005), un agent $i$ qui interagit avec l'environnement, est considéré comme un couple de systèmes dynamiques $< i, w >$, où il possède un seul état global vis à vis à la tâche de collaboration :

$$i =< P_i, Percept_i, F_i, Infl_i >$$
$$w =< E, \Gamma, \Sigma, R >$$

Un monde $w$ d'un agent est dynamique et changeant à chaque action ou réaction de l'agent collaborateur. Ce monde est modélisé par un environnement $E$, qui est l'espace dans lequel il évolue, un ensemble de répercussions $\Gamma$ causées par l'agent en question suite à la transformation de son environnement, et qui représente les trois sous ensembles des informations, des actions et des résultats, et finalement un ensemble d'états $\Sigma$ par lesquels l'agent passe, et d'une loi $R$ d'évolution du monde qu'il doit respecter pour pouvoir interagir et coopérer avec ses équivalents. Un agent $i$ est modélisé par quatre paramètres, sa fonction de perception $P_i$, son ensemble de perceptions reçues $Percept_i$, qui est l'ensemble des stimuli et des sensations qu'un agent peut recevoir, sa fonction de comportement suite à ces données $F_i$, et enfin sa fonction de production des influences suite à son comportement $Infl_i$.

### 3.1.4 Collaboration Homme-Machine-Homme *Versus* Collaboration Machine-Machine

Notre but est d'intégrer la notion de la malléabilité dans la conception de collecticiels par l'intégration ou la composition de services. Le mécanisme de l'intégration est mis en œuvre par le biais d'une collaboration avec d'autres systèmes connectés sur internet. En

fait, dans la littérature, ce type de collaboration est implémenté par une simple architecture client/serveur. Dans notre travail, notre système collabore avec d'autres machines afin d'échanger des services. Ce processus commence par une phase de communication entre les machines, puis la coordination, et finalement la phase de production, qui est l'intégration du service.

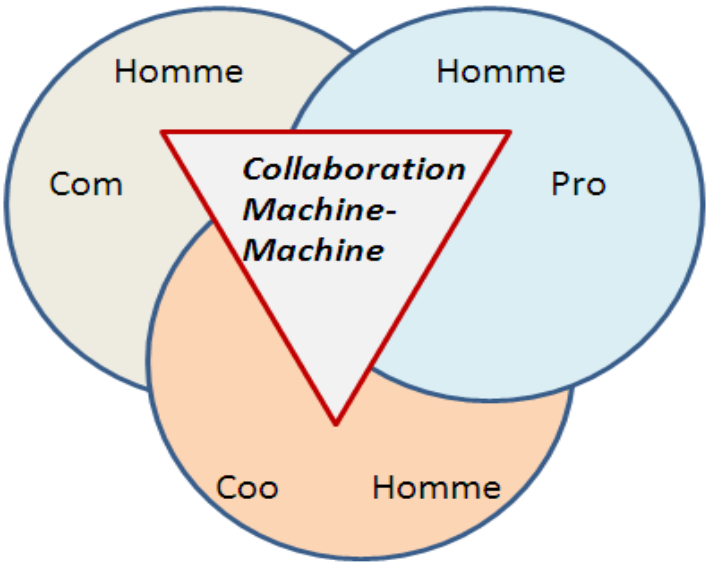



FIG. 3.3 – Collaboration Homme-Machine-Homme et Machine-Machine dans le modèle 3C

Ainsi, comme nous pouvons voir dans la figure 3.3, nous avons mis en œuvre deux types de collaboration : une "collaboration interne", que nous appelons machine-machine, et une "collaboration externe" qui représente la collaboration homme-homme. La collaboration interne intervient entre les composants d'un système ou bien entre deux systèmes connectés à un réseau afin d'assurer leurs malléabilités en échangeant des services. En effet, pour les utilisateurs en collaboration, ils ne perçoivent pas la collaboration interne, qui se fait d'une manière autonome et invisible. Ainsi, ils s'intéressent surtout aux fonctionnalités visibles et accessibles dans l'interface. Cependant, les deux types de collaboration sont fortement liés, puisque la collaboration interne génère de nouveaux services utilisables par les utilisateurs, et donc va assurer la malléabilité de la collaboration externe. Dans notre travail, nous nous intéressons à la collaboration interne. Nous argumentons qu'une collaboration marchine-machine est une première approche pour mettre en œvre la malléabilité dans la collaboration externe, et ainsi dans les logiciels de TCAO.

Pour (Khezami et al., 2005), la collaboration est un travail entre plusieurs personnes afin de produire un résultat commun (produit final). Dans notre système, cette définition correspond à la collaboration externe. En effet, la communication entre les différents membres de l'équipe est primordiale pour le succès du travail collaboratif. Dans notre système, nous supposons que la collaboration est basée sur la communication, en d'autres termes nous ne pouvons pas avoir une coordination sans communication, ni production sans coordination. Cependant, nous pouvons communiquer sans coordonner et coordonner sans produire.

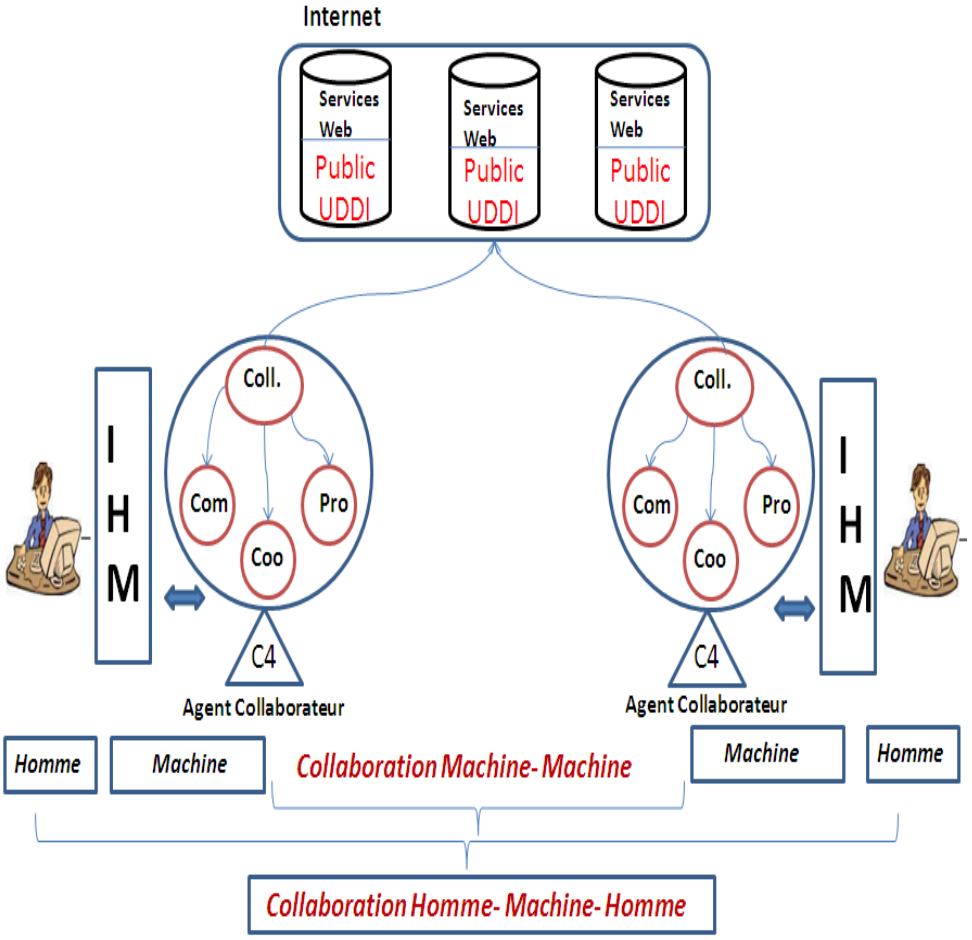



FIG. 3.4 – Collaboration Homme-Machine et Machine-Machine

En effet, toute collaboration est déclenchée suite à une demande d'un utilisateur afin de communiquer, de coordonner et de produire avec d'autres utilisateurs pour accomplir une tâche spécifique. Nous considérons qu'une collaboration inclue aussi les artéfacts et outils qui améliorent celle ci. Comme mentionné, notre objectif est d'assurer une collaboration malléable entre les utilisateurs, et ceci par l'intégration ou la composition de nouveaux services qui peuvent améliorer cette collaboration au niveau de la communication, par des chats et conférences, coordination par des outils de workflow, et de la production, qui correspond au nouveau service résultant des deux mécanismes de composition ou de l'intégration.

Ainsi, une collaboration malléable commence pour une demande d'un utilisateur de composer ou d'intégrer un service, et se termine par une collaboration machine-machine (Figure 3.4). Ce processus est mis en œuvre par un lien temporaire entre la machine locale, et les répertoires publics sur internet. Ainsi, une collaboration machine-machine est analogue à l'architecture client/serveur classique, sauf qu'elle est décomposée en trois phases : communication, coordination et production. Nous procédons par définir les terminologies utilisées dans notre formalisme de collaboration.

## 3.2 Collaboration Machine-Machine à bases de services web

### 3.2.1 Spécification et formalisme

Dans cette partie, nous présentons un formalisme pour une collaboration à bases de services web. Notre but est d'appliquer le modèle 3C afin d'introduire un formalisme de collaboration orienté services (Figure 3.5), qui soit assez générique pour pouvoir interfacer d'autres applications développées avec des standards différents. Ainsi, nous proposons l'utilisation des services web pour construire un collecticiel générique et interopérable, qui, avec leur natures distribuées et interopérables, peuvent constituer des composants de bases d'une architecture collaborative, en offrant de nouvelles fonctionnalités suite aux besoins des utilisateurs.

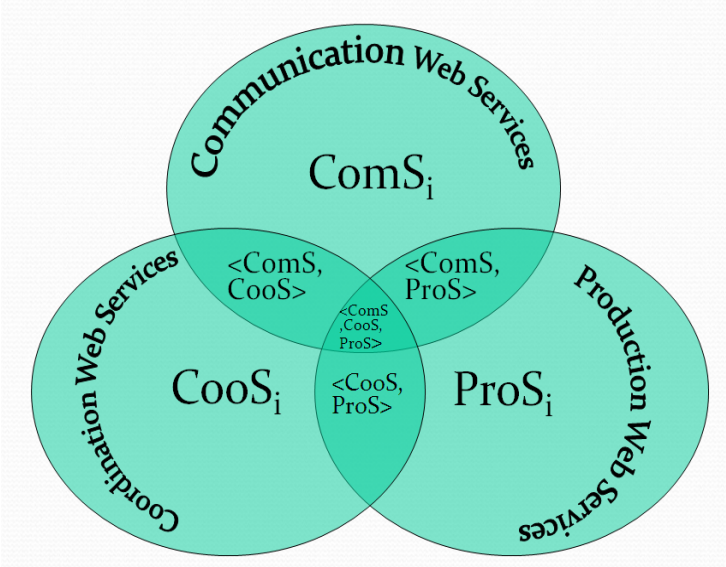



FIG. 3.5 – Interactions des trois espaces des services web

En effet, les architectures orientées-services (SOA) sont basées sur l'ingénierie des systèmes traditionnels, mais prennent en compte des caractéristiques spécifiques et surtout la collaboration, puisque les applications orientées services sont souvent collaboratives, où les consommateurs et les fournisseurs de services collaborent afin d'invoquer, de rechercher, et d'enregistrer des services. En plus, les systèmes peuvent être composés au cours de l'exécution en utilisant des services existants. Ainsi, il sera intéressant d'avoir des techniques de modélisation indépendantes de la plateforme utilisée.

#### 3.2.1.1 Terminologies

En premier lieu, nous nous basons sur les travaux de (Lee and Lee, 2007) afin de définir un service web. Nous proposons de l'utiliser et de l'étendre pour une collaboration logicielle.
Ainsi, un service web est constitué de :

$$WS =< IOPE, QoS, Loc, Coll > \tag{3.4}$$

où

$$IOPE =< input, output, precondition, effect > \tag{3.5}$$

$$QoS =< q1, q2, ..., qk > \tag{3.6}$$

$qk$ est une propriété de qualité de service, telle que la fiabilité, la disponibilité, ou le temps de réponse.

Dans notre travail, nous supposons que les services web sont localisés dans des répertoires publics (UDDI), qui peuvent être accédés avec des messages SOAP en utilisant leurs fichiers de descriptions (WSDL).

$$Loc =< UDDI1, UDDI2..UDDIk > \tag{3.7}$$

Finalement, nous supposons que les services sont classés en services de communication, coordination et production.

$$Coll =< Comm, Coor, Prod > \tag{3.8}$$

Nous pouvons voir une collaboration entre deux services $i$ et $j$ dans la figure 3.6. En effet, la collaboration nécessite au moins deux systèmes interagissant ensemble afin d'exécuter une tâche commune. Supposons qu'un service $i$ demande à un service $j$ de collaborer. Ainsi, une collaboration accomplie entre un service $i$ et un service $j$ est modélisée par une communication, une coordination et une production des deux services, comme nous pouvons voir dans l'équation 3.9.

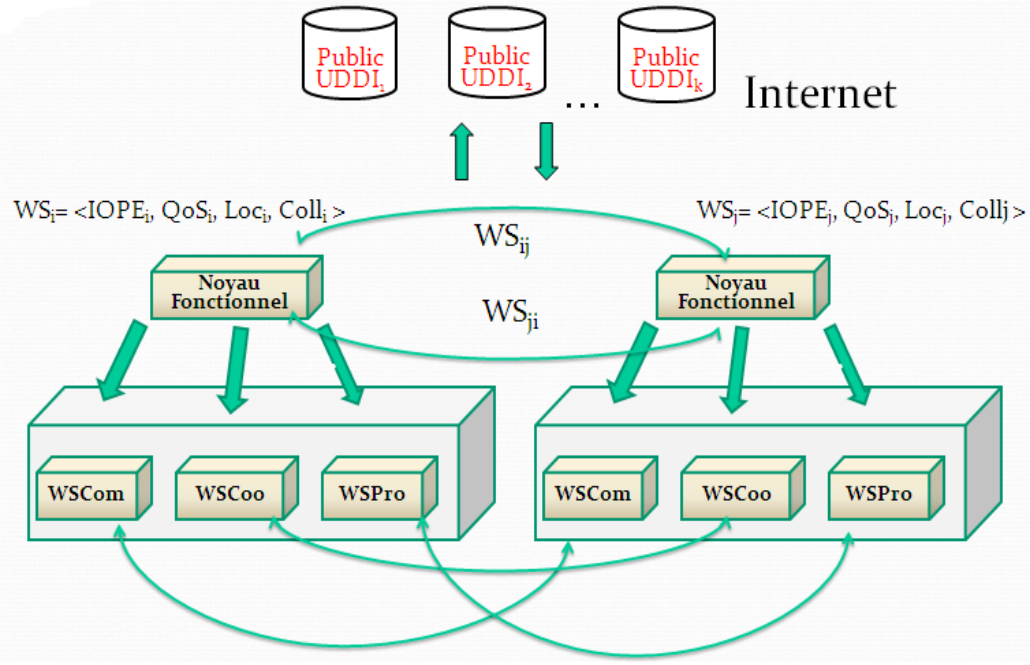



FIG. 3.6 – Collaboration à bases de services web

$$Coll^i_j =< Comm^i_j, Coor_{ij}, Prod_i, Prod_j > \tag{3.9}$$

En effet, une demande de collaboration pourra être déclenchée suite à une :

- Demande d'un consommateur ou groupe de consommateurs d'un service spécifique afin de répondre à leurs besoins.
- Demande déclenchée par un service interne afin d'être alimenté par de nouvelles ressources pour une mise à jour des fonctionnalités. Cette demande pourra inclure une procédure d'intégration ou de composition de nouveaux services.

Ainsi, nous définissons l'ensemble des actions produites par les services web, ayant pour conséquence de changer la structure interne du système. Dans notre cas, c'est l'ensemble des trois sous ensembles ($\{Inf_{ij}\}$, $\{Actions_i\}$, $\{Resultats_i\}$ :

Les $\{Inf_{ij}\}$ sont des informations échangées entre deux services $i$ et $j$, et sont :

- Informations (sémantique ou non-sémantique) sur les services internes ou qui sont susceptibles d'être intégrés dans le système, tels les IOPEs (Input, Output, Précondition, Effect), ainsi que des attributs non-fonctionnels définissant leurs qualités de service (performance, disponibilité, sécurité, etc.) et leurs localisations (si le service est externe).
- Informations relatives à la mission ou à la tâche que les utilisateurs sont en train d'effecteur. Ainsi, le système adapte les services offerts selon le contexte et la tâche.

Les $\{Actions_i\}$ sont les sous-ensembles qui contiennent toutes les actions déclenchées par les utilisateurs ou le système. Ces actions sont :

- Rechercher un service à partir de sa description syntaxique (WSDL) ou sémantique.
- Enregistrer, supprimer un nouveau service dans le système.
- Composer deux ou plusieurs services internes.
- Adapter la structure de l'application selon des descriptions de QoS et la performance des machines des utilisateurs. Ainsi, l'application offrira un ensemble de services qui correspond à la tâche, et masquera les autres services.

Pour les $\{Resultats_i\}$, c'est le sous-ensemble des résultats issus de l'exécution des actions. Un résultat est l'ensemble des actions exécutés afin de s'adapter au contexte, et pourra se déclencher suite à :

- Un nouveau service intégré dynamiquement ou statiquement (suite à une l'intervention de l'utilisateur).
- Un service composé de plusieurs services de bases.

La communication est entamée par le service $i$, d'où la notation $Comm_j^i$ ; en envoyant une demande $Inf_{ij}$ au composant principal qui gère les interactions dans le système entre les différentes couches. Ce composant est envisagé pour l'instant comme un service $j$ (équation (3.10)). Le service $j$, dans tous les cas, envoie une réponse $Inf_{ji}$ au service intiale $i$. La communication donc se repose sur un échange de couples d'information $(Inf_{ij}, Inf_{ji})$, chaque fois qu'un service entame une demande, il reçoit une réponse.

$$Comm_j^i(Inf_{ij}) = \{(Inf_{ij}, Inf_{ji})\} \tag{3.10}$$

Selon les accords des deux services $i$ et $j$, la coordination peut commencer (équation (3.11)), toujours, en discutant et en échangeant les couples d'information $(Inf_{ij}, Inf_{ji})$. Les deux services vont mettre en place des plans d'actions $(Actions_i, Actions_j)$ que chacun d'entre eux doit exécuter. Ces plans d'actions sont considérés comme des worflows à base de services web afin de coordoner les tâches.

$$Coor_{ij}\left(\{Inf_{ij}, Inf_{ji}\}\right) = \{Actions_i, Actions_j\} \tag{3.11}$$

Après la phase de coordination, commence la phase de production (équation (3.12)). Chacun des deux services, par exemple $i$ (respectivement $j$) exécute ses propres actions $Actions_i$ (respectivement $Actions_j$) et produit des résultats partiels $Resultats_i$ (respectivement $Resultats_j$).

$$Prod_i\left(\{Actions_i\}\right) = \{Resultats_i\} \tag{3.12}$$

Une fois les résultats partiels obtenus, ils sont combinés (à l'aide d'un opérateur de combinaison) dans le but d'avoir un résultat global de collaboration (un produit final).

Nous considérons la collaboration globale $COLL$ (équation 3.13) entre $N$ services, comme un triplet $< COMM, COOR, PROD >$, tel que $COMM$ est la communication globale, $COOR$ est la coordination globale et $PROD$ est la production globale de tous les services qui collaborent ensemble.

$$COLL = \langle COMM, COOR, PROD \rangle \tag{3.13}$$

La communication globale $COMM$ (équation 3.14) est représentée par tous les couples d'informations échangés entre un service $i$ et un autre $j$, pour chaque service $i$ qui communique avec un autre service $j$.

$$COMM = (Inf_{ij}, Inf_{ji}) \tag{3.14}$$

La coordination globale $COOR$ du système (équation 3.15) est représentée par l'ensemble des actions $Actions_i$ de tous les services $i$ collaborant.

$$COOR = \{Actions_i \,/ i = 1 \cdots N\} \tag{3.15}$$

La production globale $PROD$ (équation 3.16) est la combinaison de tous les résultats partiels $Resultats_i$ de tous les services $i$ collaborant. Cette combinaison produira alors soit un service "composé" à bases d'au moins deux services de bases, soit un service intégré.

$$PROD = \prod_{i=1\cdots N} (Resultats_i) \tag{3.16}$$

En conséquence, nous considérons la collaboration globale comme un processus de communication entre les services, de coordination de services et de production de services, avec comme objectifs l'échange des informations, la définition les actions à effectuer et finalement la production des résultats.

### 3.2.2 Le modèle d'architecture logicielle associé

Dans la figure 3.7, nous présentons l'architecture collaborative à bases de services web, que nous appelons *UDDI4C* ("UDDI for collaboration"). Nous nous basons sur le modèle de Arch (Bass et al., 1992) en offrant une décomposition canonique de la structure principale du système en cinq composants principaux (noyau fonctionnel, adaptateur fonctionnel, interaction physique, interaction logique et le contrôleur de dialogue), chacun ayant une fonctionnalité spécifique. Notre travail se repose principalement sur le noyau fonctionnel qui est l'élément principal du système. En plus, notre architecture se base sur le modèle conceptuel de Dewan (Dewan, 1999) qui structure le collecticiel en un nombre variable de couches. La couche la plus haute est la couche sémantique, qui correspond au noyau fonctionnel, tandis que la couche inférieure représente la couche physique ou l'interface. Dans notre travail, nous ne représentons pas les couches qui exisent entre le noyau fonctionnel et l'interface physique. Nous affirmons que ces couches peuvent contenir des modules afin de prendre en compte les points de vues des utilisateurs, ainsi que leurs profiles et l'historique de leurs utilisations des services du système.

En effet, cet environnement porte sur trois principaux aspects : (a) un cadre d'organisation des composants logiciels pour un accès à travers le réseau, (b) un mécanisme pour la publication et l'enregistrement des services afin qu'ils puissent être dynamiquement découverts, (c) un ensemble de normes qui permet aux composants d'échanger des données dans le système, ainsi qu'avec les composants distribués sur internet. Dans notre système, les services web se comportent comment un ensemble de composants de bases, pouvant interagir avec des ressources en ligne.

Au niveau plus abstrait, il existe trois composants et deux acteurs principaux : le consommateur ou un groupe de consommateurs de services qui vont interagir et collaborer en utilisant le système, et les fournisseurs qui publient leurs services dans des répertoires publics distribués sur internet, contenant des fonctionnalités divers et accessibles par des interfaces standardisés. Dans notre spécification d'environnement, comme nous pouvons le voir dans la figure 3.7, le niveau N-1 est basé sur un environnement SOA, basé sur trois composants principaux selon le modèle 3C proposé par Ellis (Ellis, 1994), qui sont décrits par une interface WSDL. Cette interface fournit tous les détails nécessaires pour interagir avec ces services, y compris les formats des messages, le protocole de transport et la localisation. Ainsi, une fois les services sont publiés, un consommateur peut trouver le service demandé via l'interface de l'*UDDI4C*. Les protocoles HTTP, SOAP et XML sont utilisés pour le transport et le triage de paramètres pour que la plateforme soit indépendante du langage de programmation, ainsi que l'accès à ces services web.

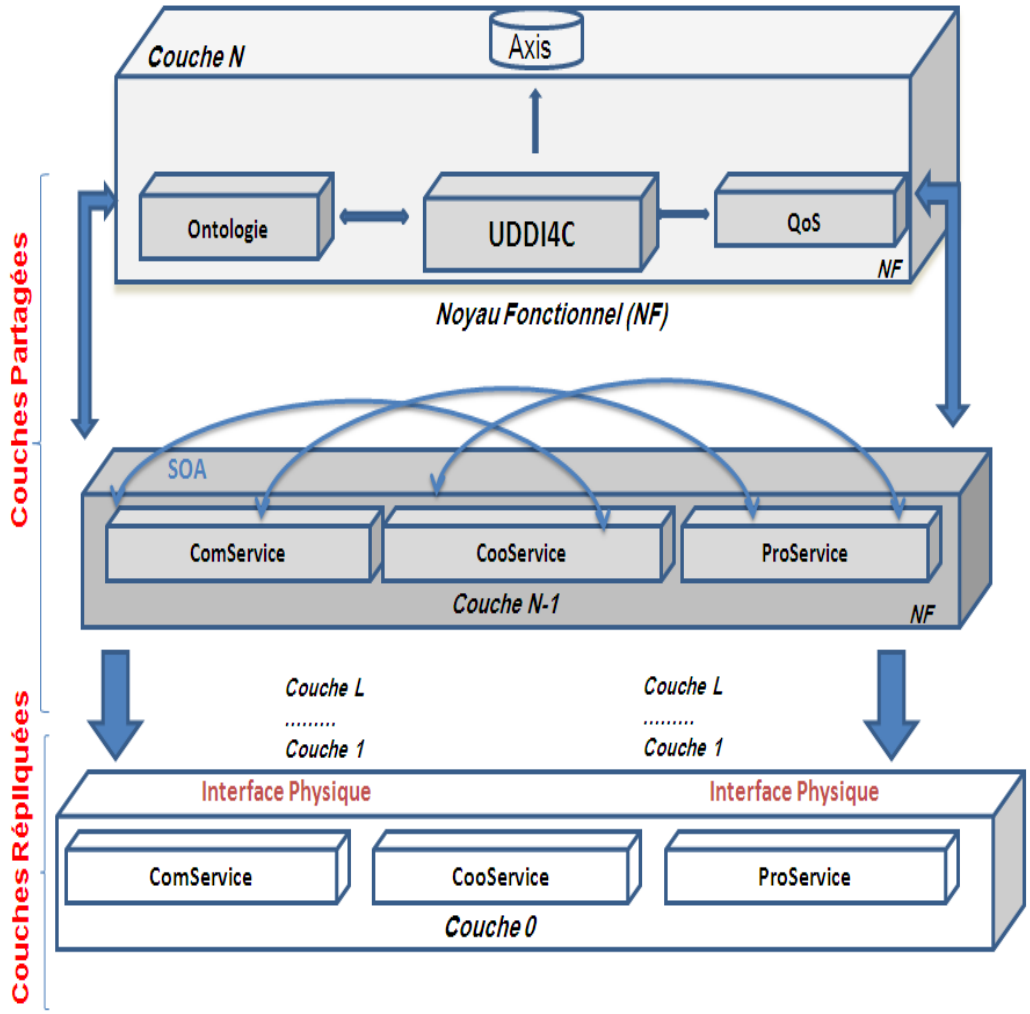



FIG. 3.7 – L'architecture collaborative à bases de services web

Avec sa nature générique, nous proposons l'architecture logicielle $UDDI4C$ comme un support à la malléabilité pour les systèmes collaboratifs à bases des services web et/ou agents logiciels. Le but est d'interfacer des systèmes "rigides" avec l'$UDDI4C$, en vue d'obtenir une certaine malléabilité au niveau des services offerts aux utilisateurs, dans un contexte collaboratif. Ainsi, cette malléabilité est obtenue par le biais de l'intégration de services web externes, ou bien de la composition de services internes afin de générer de nouveaux services de communication, de coordination ou de production. Dans le chapitre suivant, nous interfaçons l'architecture proposée avec le modèle C4, afin de calculer le coût de la malléabilité en termes d'émissions de messages dans le système. Nous rappelons que le modèle C4 (Khezami, 2005) est à la base conçu pour la téléopération d'un robot via internet. De nature rigide, le but ainsi est d'appliquer l'$UDDI4C$ sur ce modèle afin de pouvoir générer de nouveaux "services" pour la téléopération au cours de l'exécution du système.

## 3.3 Collaboration Machine-Machine à base de services web et d'agents

Le développement des logiciels a besoin d'une progression considérable afin d'exploiter de nouvelles méthodes et techniques qui soutiennent la réutilisation des composants logiciels. En d'autres termes, les développements des logiciels ont besoin de passer à des approches compositionnelles (Buhler and Vidal, 2003). Les méthodes de génie logiciel cèdent la place à de nouveaux paradigmes de développement, notamment les approches à bases de composants et d'autres approches basées-agent. ces approches gagnent une grande attention, où les composants logiciels passifs seront libérés par la dynamique et le caractère social des agents. En effet, les technologies basées-agents fournissent aux composants des mécanismes afin de s'engager dans des missions, coopérer et traiter les exigences des environnements dynamiques et hétérogènes. D'un point de vue des systèmes multi-agents (SMA), les agents ont accès à une ontologie avec une description sémantique des comportements, qui leur permet de mieux se coordonner afin de traiter des tâches spécifiques.

L'architecture orientée-services (SOA) et les systèmes multi-agents (SMA) sont deux technologies de plus en plus importantes pour la construction des systèmes logiciels. Les objectifs de ces deux architectures partagent quelques similitudes, comme par exemple, essayer de créer des systèmes distribués et flexibles qui sont composés d'entités faiblement liées, et qui interagissent les unes avec les autres. Malgré ces similitudes, il existe des différences majeures dans leurs technologies. En effet, les services possèdent des standards pour la description des interfaces ainsi que des protocoles qui sont totalement différents des langages de communications des agents. Ainsi, ils ne peuvent pas interagir les uns avec les autres directement. Certains travaux ont étudié ce problème et ont montré que l'intégration des agents et des services web produit des avantages intéressants. Les services ont une infrastructure bien définie et interopérable, alors que les agents fournissent des capacités sociales et de l'intelligence (confiance, réputation, l'engagement, etc.) pour les applications. En conséquence, l'intégration des agents et des services web améliorent l'adaptabilité, l'interopérabilité et l'ouverture du système.

Nous souhaitons introduire un formalisme pour la conception de collecticiels malléables qui intègrent ces deux approches. Ainsi, notre modèle se base sur deux environnements distincts, mais se communiquent par le biais d'une passerelle informatique. Pour l'instant, notre objectif est de montrer la décomposition des environnements des agents et services web selon le modèle 3C, définissant les trois espaces de la collaboration logicielle. Nous pouvons voir une collaboration entre le monde des agents et des services web dans la figure 3.8.

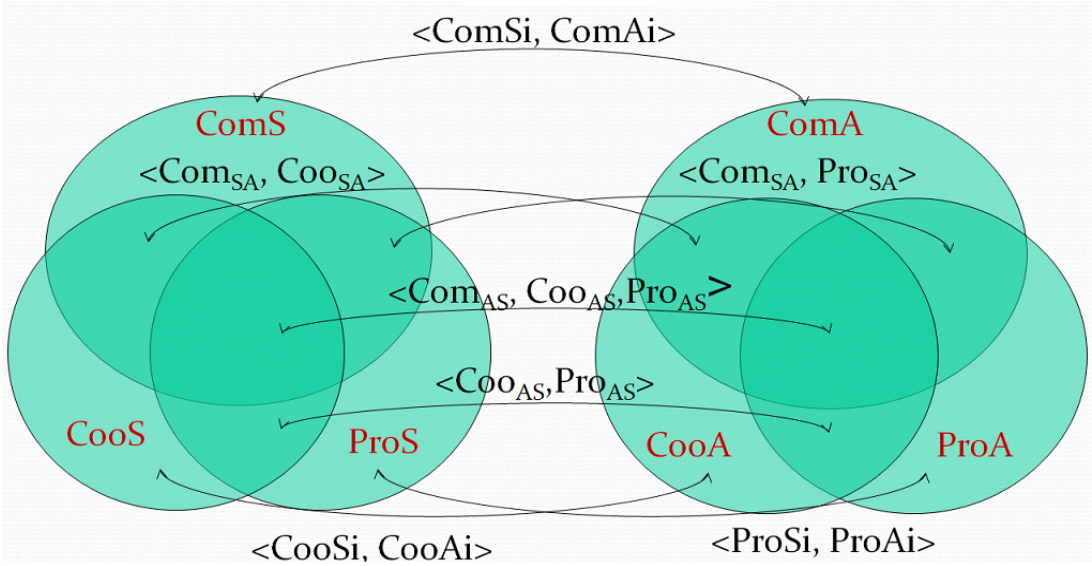



FIG. 3.8 – Collaboration entre les mondes des agents (A) et services web (S)

### 3.3.1 Spécification et formalisme

Ainsi, une collaboration malléable (Collma) entre un agent $a$ et un service $s$, est l'ensemble des interactions qui peuvent se produire entre les agents et les services web :

$$Collma(a,s) = < ComS, ComA >, < CooS, CooA >, < ProS, ProA >, < ComSA, CooSA >, \\ < ComSA, ProSA >, < CooAS, ProAS >, < ComAS, CooAS, ProAS >$$

Les agents et les services gardent leurs propriétés respectives :

$$\begin{gathered} Agent =< i, w > \\ i =< P_i, Percept_i, F_i, Infl_i > \\ w =< E, \Gamma, \Sigma, R > \end{gathered} \tag{3.17}$$

$$\begin{gathered} WS =< s, coll > \\ s =< iope, QoS, Loc > \\ coll =< comm, coor, prod > \end{gathered} \tag{3.18}$$

Comme nous pouvons le constater, les deux mondes se communiquent par le biais de messages interopérables afin qu'ils puissent échanger des informations. En effet, pour que nous arrivions à une collaboration malléable, les trois espaces (communication, coordination et production) des deux environnements (SOA et agents) doivent interagir ensemble.

#### 3.3.1.1 Agent de collaboration hybride - Orchestration de services

En effet, nous ne voulons pas implémenter les services web comme étant des agents intelligents eux mêmes, ce qui pourra rompre la compatibilité avec les services web existants. Dans notre travail, les services web sont considérés comme des ressources afin d'être

exploiter par des agents logiciels. Ainsi, nous gardons l'autonomie des deux mondes tout en bénéficiant de leur synergie pour concevoir une architecture collaborative. Vu que ces deux technologies utilisent des protocoles et des standards différents, leur intégration doit être gérée par un composant spécial qui assure leurs communications. Notre choix d'implémentation est justifié dans le chapitre suivant.

Dans la figure 3.9, les services sont divisés en trois parties : communication, coordination et production. Ces espaces sont considérés comme un ensemble de services de bases offrant des fonctionnalités relatives à leurs trois espaces. Au niveau conceptuel, un agent spécial que nous appelons WAG ("Web AGent"), gère les interactions entre les services web et les agents dans le système. Cet agent est ainsi considéré comme un agent hybride, doté de certaines caractéristiques spéciales comme la proactivité, afin de pouvoir interagir d'une façon autonome avec d'autres composants. Le but est d'assurer "l'orchestration" des services afin de les d'intégrer à partir de leurs descriptions fonctionnels et non-fonctionnels, et de les composer avec des services existants dans le système. Ainsi, le but est de définir des processus de workflow pour la gestion des données et des services.

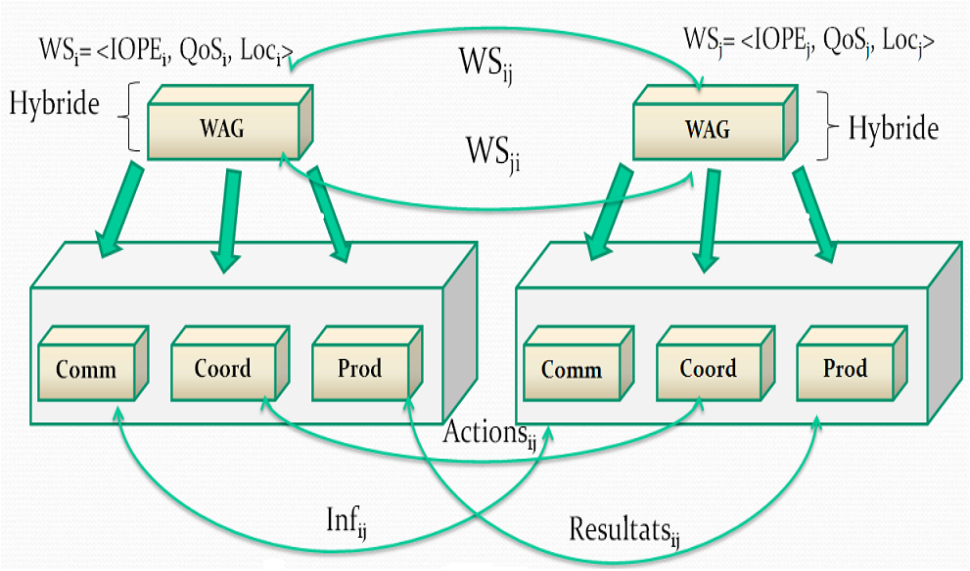



FIG. 3.9 – WAG - Agent Hybride

Nous pouvons voir dans la figure 4.17 le processus d'orchestration afin d'intégrer un nouveau service web, en prenant en compte sa qualité de service. Ainsi, une collaboration entre deux systèmes $i$ et $j$ commence pour un échange (synchrone) d'information sur le service à intégrer (à partir de son fichier de description WSDL et sa QoS), qui constitue la phase de communication. Une fois terminée, la phase de coordination commence, où les informations relatives au service sont envoyées aux agents de coordination. Finalement, Les agents de production extraient le service et l'enregistre dans le système. Ainsi, la QoS de service est recalculé, ainsi que les informations relatives à sa localisation.

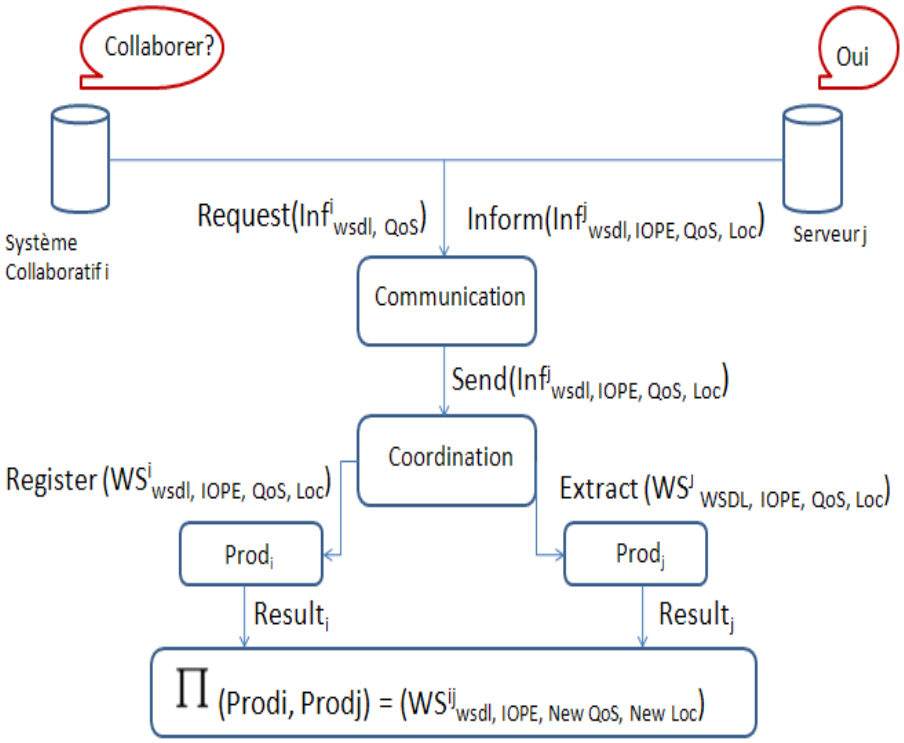



FIG. 3.10 – Protocole d'orchestration afin d'intégrer un nouveau service

Dans la figure 3.11, nous pouvons voir le processus d'orchestration afin de composer deux services internes. Ainsi, la phase de communication commence par un échange d'informations (notamment des informations relatives à leurs descriptions syntaxiques et sémantiques) entre deux services $i$ et $j$ par le biais du chef d'orchestre. Une fois terminée, la phase de coordination commence qui teste la faisabilité de la composition, où elle aura lieu si le service composé un nouveau comportement. Finalement, la phase de production commence, qui va produire le nouveau service, en l'enregistrant dans le système, et en redéfinissant sa description syntaxique (WSDL) et son IOPE (par le biais de son ontologie correspondante), alors que sa QoS est l'addition des deux QoS des services initiaux.

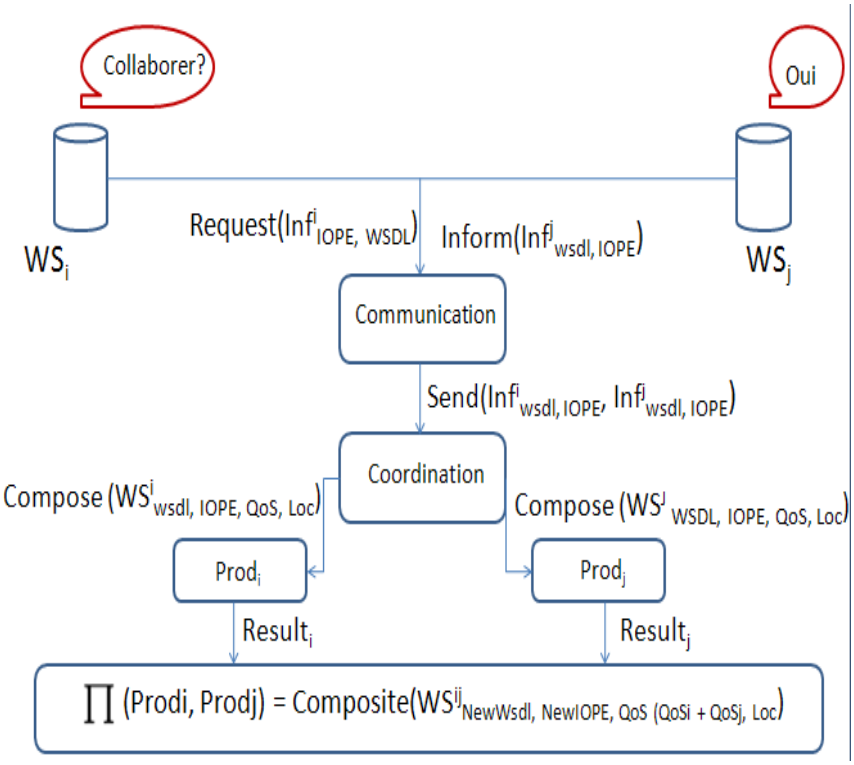



FIG. 3.11 – Protocole d'orchestration afin de composer un service à partir de deux services (Nous supposons que ces deux services sont de la même classe (communication, coordination et production)

Dans le chapitre 4, nous mettons en œuvre une étude afin de mesurer la performance du processus d'orchestration. Le but est de comparer les résultats qui peuvent en résulter avec le modèle C4 (Khezami, 2005), afin de démontrer l'intérêt de cette approche pour mettre en œuvre la malléabilité dans notre architecture collaborative.

#### 3.3.1.2 Agents et services de communication

Comme mentionné précédemment, le but d'appliquer le modèle 3C est d'assurer la modularité du système qui, éventuellement, réduit la complexité d'implémentation. Par exemple, ça sera plus simple d'ajouter un service de communication (vidéo, chat, etc.) sans affecter les autres services. Les services de communication sont gérés par les agents de communication, qui gèrent toutes les informations qui lui sont communiquées. Ces informations représentent les différentes perceptions qu'ils peuvent recevoir ($Percept_j$), comme celles émises par l'agent WAG ($Inf_i$) ou d'autres agents dans le système. L'ensemble des entrées et des sorties de l'agent de communication sont :

- $\{Inf_{ij}\}$ : l'ensemble des informations envoyé par l'agent de communication $i$ vers les autres agents de communication $j$ avec lesquels il est en collaboration.
- $\{Percept_i\}$ : l'ensemble des stimulis et des sensations qu'il peut produire.

#### 3.3.1.3 Agent et services de coordination

La coordination est une étape fondamentale avant de produire. Ainsi, l'agent de coordination définit les actions avec le WAG afin de produire un nouveau comportement,

soit par la composition soit par l'intégration de services. Ainsi, cet agent accomplit les actions suite au choix de la mission de l'agent communication. L'agent de coordination reçoit un ensemble d'informations qui lui sont envoyées, et qui regroupent les différentes perceptions ($Percept_j$), ainsi que les informations qui lui sont envoyées par l'agent WAG ($Inf_i$). Comme pour l'agent communication, l'ensemble des entrées et sorties de l'agent coordination sont :

- $\{Actions_i\}$ : l'ensemble des actions exécutées par l'agent $i$. Dans notre système, les deux tâches de coordination les plus importantes sont celles définit par le processus d'orchestration : l'intégration de nouveaux services, et la composition de deux ou plusieurs services web internes.
- $\{Percept_i\}$ : Ces stimulis sont les paramètres fonctionnels (IOPE) et non-fonctionnels (QoS) que les services web peuvent produire.

#### 3.3.1.4 Agent et services de production

La production est le résultat de la collaboration entre utilisateurs. Ainsi, l'agent de production se charge de l'exécution des actions issues de la phase de coordination et envoyées par l'agent WAG. Comme pour les deux autres agents, l'ensemble de ses entrées et sorties sont :

- $\{Percept_i\}$ : l'ensemble des stimuli et des sensations qu'il peut produire.
- $\{Resultats_i\}$ : l'ensemble des résultats issus de l'exécution des procédures associées à ses actions. Ces résultats contiennent le nouveau service produit, ainsi que toutes les informations relatives à ce service, à noter son adresse, ses informations sémantiques et sa QoS.

### 3.3.2 Le modèle d'architecture logicielle : $U3D$

Nous rappelons que notre objectif est d'intégrer les agents logiciels et les services web, en une entité cohérente qui tente de dépasser la faiblesse de chaque technologie, tout en renforçant leur avantages individuels. Dans nos travaux (Cheaib et al., 2008b), (Cheaib et al., 2010), nous avons proposé l'utilisation de l'architecture orientée-services pour la conception des collecticiels malléables, qui offre l'interopérabilité et la re-configurabilité nécessaire des composants du système. Nous avons aussi discuté l'importance d'utiliser des agents logiciels afin d'améliorer la découverte proactive des services.

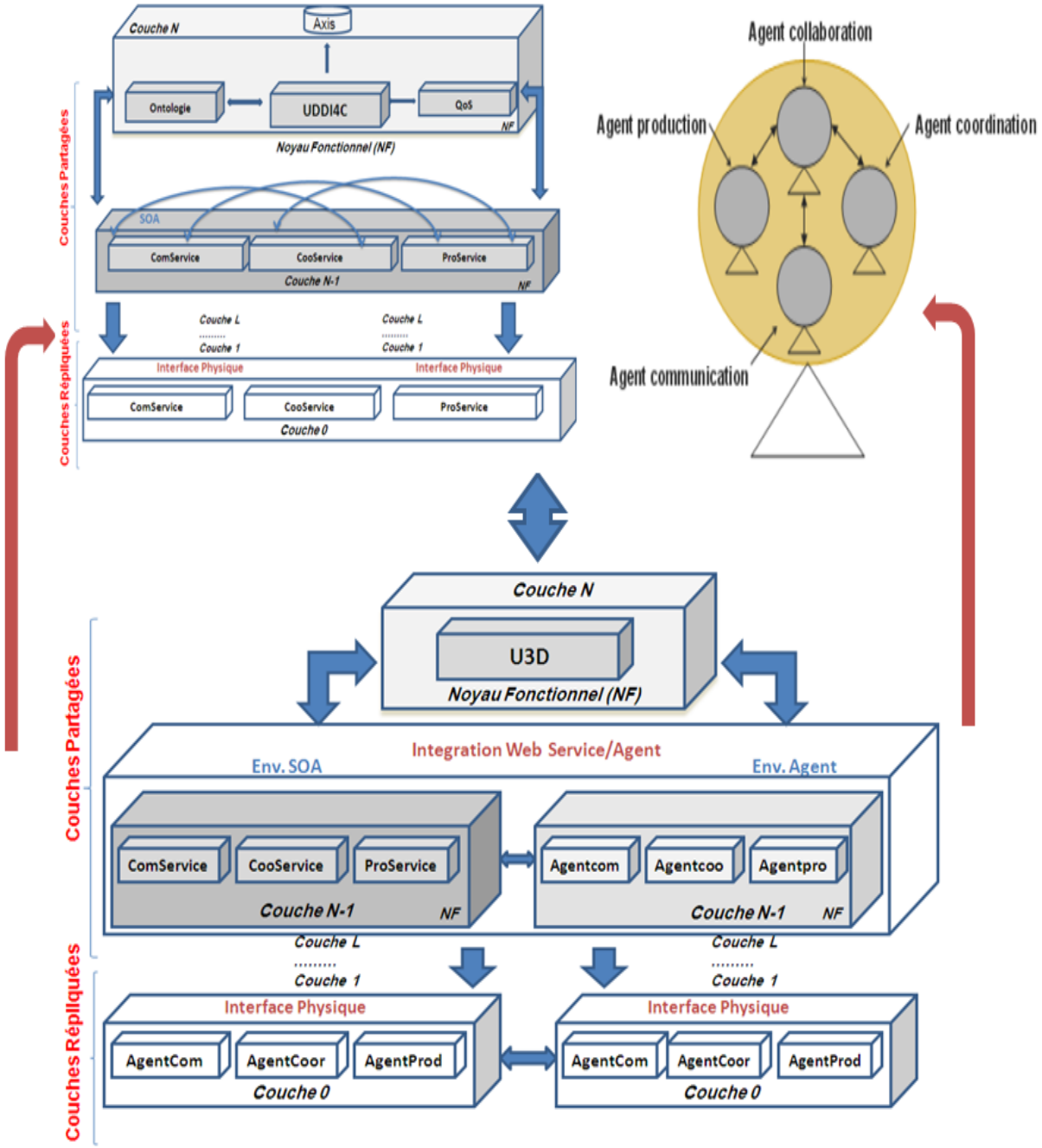



FIG. 3.12 – L'architecture U3D

Nous pouvons voir l'architecture globale dans la figure 3.12. Cette nouvelle architecture est conçue à partir de l'intégration de l'agent C4 (Khezami et al., 2005), et de l'architecture *UDDI4C* proposée dans la section précédente. Ainsi, nous obtenons une architecture composée de deux environnements parallèles, travaillant de concert afin de mettre en œuvre une architecture logicielle collaborative et malléable, bénéficiant des propriétés des deux technologies. Toutefois, notre noyau fonctionnel est divisé en deux couches publiques, et permet à tous les utilisateurs de manipuler des objets du domaine et avoir accès à divers services au cours de l'interaction avec le système. Ce mécanisme est différent du modèle clover (Laurillau, 2002), qui divise le noyau fonctionnel en deux couches, une publique et l'autre privé pour chaque utilisateur. Dans notre modèle, toutes les autres couches sont répliquées, et gèrent l'ensemble des services pour chaque utilisateur. Il faut noter que la figure 3.12 ne montre que le noyau fonctionnel ainsi que la couche

d'interaction physique. Dans la dernière section, nous mettons en œuvre la composante physique comme étant une interface web qui servira comme une étude de cas de notre modèle.

Nous étendons cette abstraction de couches comme dans (Laurillau, 2002), par la décomposition de chaque couche de l'architecture en sous-composants. Chaque couche est dédiée à un aspect du modèle 3C, en assurant la gestion des services spécifiques pour la communication, la coordination et de la production. Cependant, nous supposons que seulement les couches sur le niveau N-1 et sur le niveau le plus bas (niveau 0) mettent en œuvre cette décomposition, tandis que nous ne faisons aucune hypothèse sur la décomposition des autres couches. Les sous-composants sur le niveau N-1 sont enfermés dans une interface logicielle exposant ses fonctionnalités aux clients.

#### 3.3.2.1 Environement SOA

Comme nous pouvons voir dans la figure 3.12, une partie du niveau N-1 est basée sur un environnement SOA en charge des services web dans le système, où nous retrouvons des services pour la communication, la coordination et la production comme suit :

- ComService : les services offrant des moyens de communication entre les utilisateurs en collaboration (service de vidéoconférence, audio, communication textuelle etc.)
- CoorService : les services implémentant des règles de coordination. Normalement, ces types de services sont mis en œuvre en utilisant des techniques de workflow afin de codifier leurs interactions.
- ProService : les services qui sont le produit de la collaboration entre les utilisateurs (application Paint, document Word, etc.), ainsi que tous les services qui offrent des fonctionnalités d'aide à la production.

Ces trois groupement de services sont considérés comme une orchestration de divers services (Peltz, 2003a), fournissant des fonctionnalités de la même classe (communication, coordination et production).

#### 3.3.2.2 Environnement d'agents

En parallèle, un environnement d'agents constitue l'autre partie du noyau fonctionnel (NF) au niveau N-1. Cette couche est peuplée d'agents logiciels qui sont déployés en utilisant des bibliothèques pour mettre en œuvre les comportements d'agents. Le paradigme de la communication entre les couches se fait par un échange asynchrone de messages. L'utilisation des agents, comme nous l'avons mentionné précédemment, est de rendre la découverte de nouveaux services dynamique, sans l'arrêt de l'exécution du système. La décomposition fonctionnelle de cet environnement selon le modèle 3C met en œuvre une grande modularité, où chaque agent gère un ou plusieurs services dans l'environnement SOA de la même classe. Chaque sous-composant de cette couche manipule des objets sémantiques dédiés à l'une des trois facettes du modèle 3C, et exerce des fonctions de traitement spécifiques sur ces services.

#### 3.3.2.3 Universal Directory for Description and Discovery

Nous décrivons la couche au niveau N qui constitue la couche sémantique de l'architecture, et qui est la composante de base dans le modèle. En effet, le nom du modèle est inspiré de l'intégration de l'UDDI (Universal Description Discovery and Intégration) utilisé par les services web, et le DF (Directory Facilitator) qui est analogue au UDDI, utilisé par les agents logiciels en utilisant la plateforme JADE[1]. Dans le chapitre 4, nous présentons en détails les interactions entre les agents ainsi que la platforme de dévelopement associée. Ainsi, nous obtenons le "Universal Directory for Description and Discovery", ou l'*U3D*. Nous visualisons un scénario où un agent recherche un service dans le système. Le modèle déclenche un mécanisme pour rechercher la disponibilité des services dans l'environnement SOA. Si le service web est trouvé, l'agent l'enregistre en tant que sont propre service.

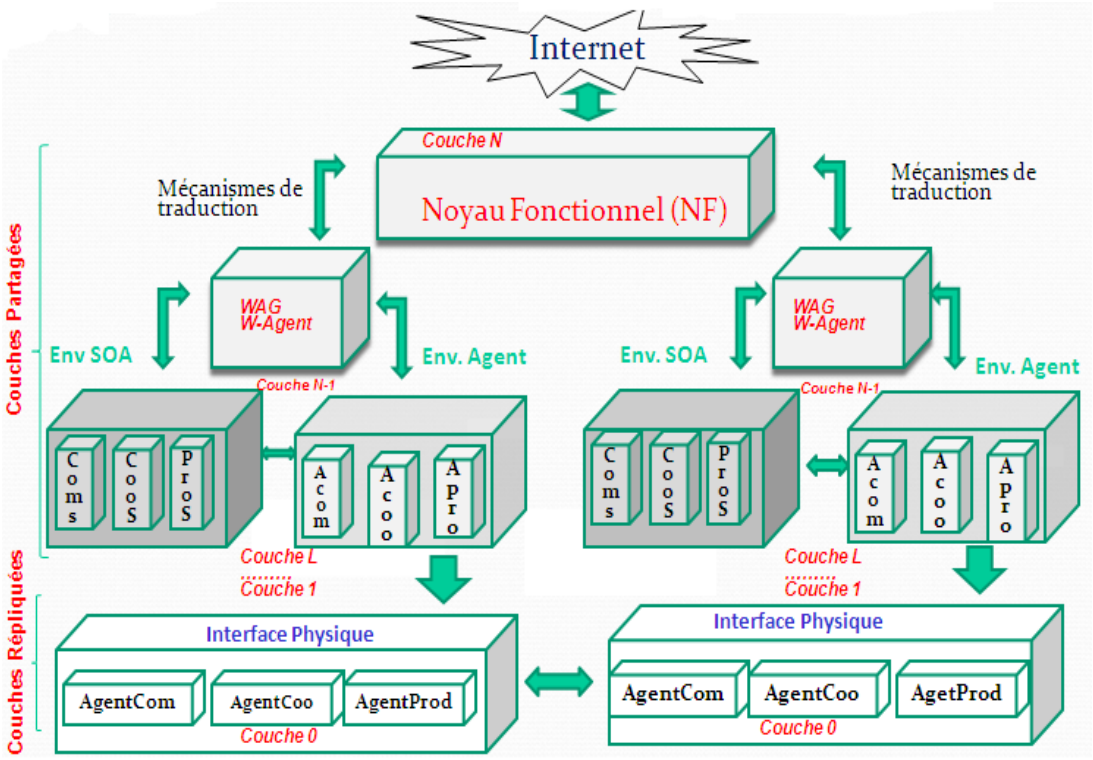



FIG. 3.13 – Interactions inter-composants dans l'architecture globale

Toutefois, l'UDDI utilisé n'est pas capable de stocker et de traiter des descriptions sémantiques. En conséquence, il est clair qu'un registre qui prend en charge l'annotation sémantique et la mise en correspondance, produira une recherche encore plus précise, ainsi qu'un meilleur mécanisme d'invocation et d'intégration. Il sera nécessaire d'utiliser des métadonnées sémantiques que les agents logiciels puissent comprendre et interpréter, avec une intervention humaine minimale. En conséquence, un langage spécifique devrait être utilisé pour rajouter de la sémantique pour les valeurs des entrées et sorties du service web, en joignant des métadonnées qui référencent des concepts d'ontologies. Les agents peuvent alors comprendre les descriptions du service, permettant une intégration et une composition dynamique. Dans la figure 3.13, nous présentons les interactions entre les composants au sein de l'architecture. Le WAG gère les liens entre les agents et services au sein du système, ainsi que les agents et les services web externes. Ainsi, il est situé au

[1]http ://jade.tilab.com/

plus haut niveau sémantique de l'architecture.

En conséquence, ce composant est mis en œuvre pour que :

- Les agents logiciels puissent découvrir et invoquer des services web.
- Les clients des services web découvrent et invoquent les services d'agents dans le Directory Facilitateur (DF) de l'environnement JADE.
- Les services web soient publiés dans le DF, comme étant des services d'agents.

#### 3.3.2.4 Discussion

L'originalité de notre modèle est l'utilisation des technologies existantes afin de créer une architecture logicielle malléable pour les collecticiels, où un tel modèle n'est pas encore bien identifié dans la littérature. En plus, ce modèle est inspiré des modèles d'Arch et de Dewan pour séparer les fonctionnalités de base (la logique de l'application) de leurs interfaces, apportant de nombreuses propriétés essentielles telles que la flexibilité qui est cruciale dans le domaine de la collaboration. Toutefois, les deux couches constituant le noyau fonctionnel sont partagées et gèrent exclusivement les services, ainsi que leurs intégrations et leurs compositions. Ce mécanisme est différent du modèle clover, qui préconise une réplication d'une partie du noyau fonctionnel pour chaque utilisateur. Dans notre modèle, l'adaptateur du noyau fonctionnel qui est situé entre la couche physique et le noyau fonctionnel (que nous n'avons pas abordé dans notre travail), est plus apte à gérer ce type de données. Ainsi, nous consacrons le noyau fonctionnel pour gérer uniquement les services web afin d'assurer la malléabilité nécessaire. En conséquence, chaque nouveau service sera partagé par tous les utilisateurs qui participent en collaboration à une session de travail.

La répartition fonctionnelle selon le modèle 3C d'Ellis possède plusieurs propriétés. En fait, du point de vue implémentation, cette division fonctionnelle se traduira par une plus grande modularité, qui permet de réduire la complexité de l'implémentation. En outre, nous n'avons pas mis d'hypothèse sur les autres couches du système par rapport à la décomposition fonctionnelle selon de modèle 3C, ce qui réduit les coûts de développement et le temps de calcul, en permettant l'ajout des couches indépendantes et hétérogènes pour améliorer la distribution des fonctions, et en assurant l'interopérabilité sur chaque couche de l'architecture (en utilisant les standards FIPA et les spécifications du W3C).

Pour le point de branchement discuté dans (Patterson, 1995), nous l'avons fixé après les deux premières couches (N et N-1) qui constituent le noyau fonctionnel. Cela induit un taux de réplication plus petit que le modèle clover, mais plus convenant afin d'assurer la cohérence de l'état des services et leurs réutilisations. Enfin, notre modèle identifie l'architecture d'implémentation qui est déduite du modèle théorique en vue d'implémenter explicitement la malléabilité. Contrairement à d'autres modèles, nous identifions explicitement le composant comme un service web et un agent collaborant ensemble pour offrir des services unifiés. Dans notre modèle, les agents mettent en œuvre le concept du code mobile, où leurs coordinations avec leurs architectures flexibles leur permettent de s'adapter à des environnements dynamiques et hétérogènes tel que le web.

## 3.4 Bilan

A partir de l'étude des services web et des agents logiciels, nous avons proposé deux formalismes avec leurs architectures associées. Le premier formalisme met en œuvre une architecture à bases de services web. Le deuxième formalisme prend en compte les caractéristiques des systèmes multi-agents (SMA), des services web et de la collaboration malléable comme suit :

1. Les propriétés des SMA :
   - *La communication* : les agents communiquent ensemble pour trouver un arrangement. Cette propriété est commune au SMA et aux systèmes de collaboration.

   - *L'interaction* : les agents ont un protocole d'interaction bien défini. Si un agent reçoit une information d'un autre agent, il est dans l'obligation de répondre.

   - *La coordination* : les agents se coordonnent pour collaborer. Cette propriété, aussi, est commune.

   - *L'intelligence sociale* : les agents sont autonomes, chacun d'entre eux prend une décision (de coordonner ou pas) selon les informations dont il dispose.

2. Les propriétés des services web :
   - *Interopérabilité* : Les standards utilisés par les services web (WSDL, SOAP et UDDI) sont interopérables, ainsi l'utilisation des services web est indépendante de la langage de programmation ainsi que le système d'exploitation utilisé.

   - *Réutilisation* : Avec leur nature générique en tant que des composants logiciels, les services web favorisent leurs réutilisations afin d'éviter de reprogrammer de nouveaux composants qui peuvent offrir les mêmes fonctionnalités. Ainsi, ils minimisent le coût d'implémentation pour les développeurs.

   - *Intégration* : Les services web favorisent leurs intégrations à partir de leurs fichiers de descriptions WSDL, qui publient les opérations offertes par le service en XML. Ainsi, un agent peut choisir le service à utiliser sans pour autant avoir une grande connaissance sur l'implémentation de bas niveau du service, facilitant sa découverte.

3. Les caractéristiques de la collaboration :
   - *La communication* : les agents communiquent pour choisir une mission. c'est une propriété commune avec les SMA.

   - *La coordination* : c'est une propriété commune avec les SMA.

   - *La production* : les agents quand ils s'engagent dans un processus de collaboration et qu'ils définissent des plans d'actions, ils sont obligés de mener au bout leurs exécutions pour produire un résultat partiel. L'ensemble de tous les résultats partiels combinés forme un résultat global et ainsi un produit final de la collaboration.

4. Les propriétés de la malléabilité :
   - *L'adaptabilité* où les services offerts aux utilisateurs sont adaptés selon le contexte et la tâche à effectuer.

   - *La composition* qui permet de créer de nouveaux comportements à partir de services web de base comme étant des composants logiciels. Ainsi, les logiciels construits à base de composants et dont la structure de composition est maintenue visible, peut être modifiée, tout au long de l'utilisation du logiciel.

   - *L'ouverture* qui permet d'accueillir de nouvelles formes de structures de données et de contenus, ainsi que de nouveaux outils collaboratifs.

## 3.5 Conclusion

Dans ce chapitre nous avons proposé, dans la première section, un formalisme pour la collaboration machine-machine à base de services web. Ensuite, nous avons décris l'architecture logicielle associée, le modèle $UDDI4C$ qui agit comme un support de malléabilité pour les systèmes à bases de services web et/ou agents logiciels. Dans la deuxième section, nous avons proposé un formalisme pour la collaboration machine-machine malléable à bases de services web et des agents logiciels. Ces derniers sont inspirés du formalisme du SMA-C, qui est à la base du modèle C4 (Khezami et al., 2005). Ensuite, nous avons décris l'architecture logicielle associée, l'architecture $U3D$.

En effet, notre but est d'étudier un nouveau mode de communication entre ces deux technologies afin d'améliorer le processus de collaboration. Dans leurs natures, les services web sont une technologie qui peut être appliquée et intégrée avec d'autres technologies afin d'accroitre leurs potentiels de découverte, d'intégration et d'exploitation. La recherche sur les systèmes multi-agents qui se relient aux technologies de l'internet, se concentre sur la construction des systèmes définissant la théorie derrière les comportements des agents, ainsi que la communication inter-agents. La plupart des systèmes qui en résultent sont des systèmes fermés qui fonctionnent bien au sein d'un organisme ou un environnement homogène. En revanche, ces travaux ont mi moins l'accent sur l'interopérabilité des agents dans un monde hétérogène, et en perpétuelle évolution, comme le web. En plus, les systèmes à bases d'agents qui fonctionnent au delà des frontières d'un organisme, nécessitent de résoudre plusieurs contraintes dans la découverte des services, ainsi que pour la communication et la réutilisation.

Dans le chapitre suivant, nous décrivons les technologies utilisées pour la mise en œuvre de notre architecture logicielle $U3D$. Ensuite, nous proposons une étude en UML ainsi qu'une mise en œuvre des interactions entre les différents composants de l'architecture logicielle. Finalement, nous décrivons une analyse qui démontre une meilleure performance par rapport au modèle C4 (Khezami, 2005) en termes de nombres de messages émis dans le système.

# Chapitre 4

# CONCEPTION ET IMPLEMENTATION

L'intégration des agents logiciels et des services web doit être mise en œuvre au niveau de la conception et de l'implémentation. Au niveau de la conception, les services sont encapsulés comme étant des services d'agents, de sorte que chaque agent fonctionne dans son action et en relation avec l'environnement au nom d'un service web. Ainsi, ces services peuvent être employés pour décrire les comportements externes des agents et résoudre le problème d'interopérabilité des ressources, alors que les agents, eux, sont utilisés pour établir un modèle d'interaction flexible de haut-niveau. Au niveau de l'exécution, UDDI, WSDL et SOAP fournissent des capacités telles que la découverte, le déploiement et la communication, alors que des langages de spécification telles que BPEL4WS fournissent le processus de la composition.

Notre chapitre est divisé en deux grandes parties. Dans la première section, nous décrivons les plateformes et technologies utilisées pour mettre en œuvre notre architecture logicielle $U3D$. Nous présentons une étude en UML, en définissant les classes principales du système. Puis, nous proposons la mise en oeuvre des interactions entre les couches du noyau fonctionnel (NF) du l'$U3D$. Ces interactions s'appliquent à l'$UDDI4C$ dans le cas où ce modèle est interfacé avec un système multi-agents. Dans la deuxième section, nous proposons une étude des interactions dans les deux architectures logicielles proposées, où nous prouvons que les résultats attendues sont meilleurs par rapport aux résultats issues de l'étude du modèle C4 (Khezami, 2005). L'étude présentée ne gère pas les utilisateurs humains, mais tout simplement la performance du système en termes de nombre de messages envoyés dans le processus de collaboration.

## 4.1 Outils et Technologies utilisés

Avant de présenter la mise en oeuvre de l'architecture logicielle, nous commençons par présenter l'outil de conception UML qui nous a servi pour représenter les principales classes des agents du système. Puis, nous présentons la plateforme JADE pour la mise en œuvre de ces agents, ainsi que le WSIG, qui est un composant de JADE.

### 4.1.1 UML

UML (Unified Modeling Language) est né de la fusion des trois méthodes qui ont influencé la modélisation objet au milieu des années 90 : OMT, Booch et OOSE. Issu d'un travail d'experts reconnus, UML est le résultat d'un large consensus. UML comble une lacune importante des technologies objet, en fait il permet d'exprimer et d'élaborer des modèles objet, indépendamment de tout langage de programmation, il a été pensé pour servir de support à une analyse basée sur les concepts objet. C'est un langage formel, défini par un métamodèle, il permet de représenter un système selon différentes vues complémentaires, à savoir, les diagrammes. Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle ; c'est une perspective du modèle, chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis) et véhicule une sémantique précise (il offre toujours la même vue d'un système). Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système. Ce langage opte en effet pour l'élaboration des modèles, plutôt que pour une approche qui impose une barrière stricte entre analyse et conception. Les modèles d'analyse et de conception ne diffèrent que par leur niveau de détail, il n'y a pas de différence dans les concepts utilisés. Il n'introduit pas d'éléments de modélisation propres à une activité (analyse, conception, etc.). Ainsi, ce langage reste le même à tous les niveaux d'abstraction (Khezami, 2005).

### 4.1.2 JADE

Le Java Agent DEvelopment framework[1] (JADE) est un middleware écrit en Java et conforme aux spécifications de la FIPA. Cet environnement simplifie le développement d'agent en fournissant les services de base définis par la FIPA, ainsi qu'un ensemble d'outils pour le déploiement. La plateforme JADE peut être répartie sur un ensemble de machines et configurée à distance, grâce à un mécanisme de migration d'agent au sein de la même plateforme. La plateforme JADE contient :

- Un environnement d'exécution, où les agents JADE peuvent évoluer. Il doit être actif sur un hôte donné et ainsi un ou plusieurs agents peuvent être exécutés sur cet hôte.
- Une librairie de classes que les programmeurs utilisent pour développer leurs agents.
- Une suite d'outils graphiques qui permettent l'administration et la supervision des activités des agents en exécution.

Le conteneur principal contient deux agents spéciaux :

- L'AMS (Agent Management System) qui fournit un service de nommage (i.e. il assure que chaque agent dans la plateforme possède un nom unique). Il représente l'autorité dans la plateforme (c'est possible de créer ou de tuer des agents dans des conteneurs distants en le demandant à l'AMS).
- Le DF (Directory Facilitator), offre le service de Pages Jaunes au moyen duquel un agent peut trouver d'autres agents fournissant les services dont il a besoin dans le but d'atteindre son objectif.

[1] http ://jade.tilab.com/

Le DF permet aux agents de publier des descriptions d'un ou de plusieurs services qu'ils fournissent afin que d'autres agents peuvent facilement les découvrir et les exploiter. En effet, n'importe quel agent peut à la fois enregistrer (publier) ou rechercher (découvrir) des services. Le DF est conforme avec les spécifications de la FIPA, où chaque plateforme conforme à cette norme devrait avoir un DF par défaut. En effet, un agent qui désire publier un ou plusieurs services, doit fournir au DF une description qui inclut son AID (Identifiant de l'agent), une liste des services fournis et, éventuellement, la liste des langues et des ontologies que d'autres agents doivent utiliser pour interagir avec l'agent concerné. Le `DFAgentDescription`, `ServiceDescription` et les classes de propriétés inclus dans le `jade.domain.FIPAAgent` représentent ces abstractions.

Nous avons interfacé le DF avec La librairie JUDDI [2], qui offre une interface afin d'interagir avec l'UDDI du système. En effet, il existe plusieurs technologies afin d'implémenter les services web, mais dans notre travail, nous avons utilisé les protocoles standards (UDDI, SOAP et WSDL) pour leurs larges utilisations dans le monde industriel.

En effet, un dispositif clairement utile de JADE serait des moyens de rechercher et d'appeler des services directement comme des agents d'application. Le WSIG ('Web Service Integration Gateway') satisfait ces besoins en fournissant les moyens d'enregistrer des services dans l'environnement JADE.

### 4.1.3 WSIG

Le WSIG [3] est un composant de la plateforme JADE, qui tente de traduire les messages utilisés par les services web, à des messages compréhensibles par les agents logiciels. Ainsi, il supporte la pile des protocoles standards des services web (WSDL, SOAP et UDDI). En effet, comme nous pouvons voir dans la figure 4.1, le WSIG est une application web composée de deux éléments principaux, le "WSIG servlet" et le "WSIG agent". Le "WSIG servlet" représente la facette vers le monde d'Internet et est responsable de :

[2]http ://ws.apache.org/juddi/
[3]http ://jade.tilab.com/

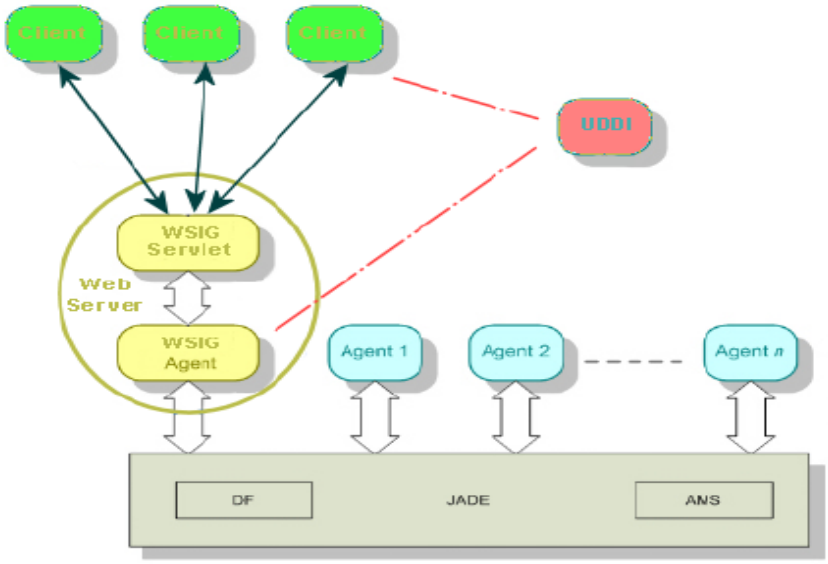



FIG. 4.1 – Capture d'ecran du WSIG

- Collecter les demandes d'entrées HTTP / SOAP demandes
- Extraire le message SOAP
- Préparer l'action d'agent correspondant et le transmettre au "WSIG Agent". Une fois l'action est faite :
- Convertir le résultat de l'action à un message SOAP.
- Préparer les protocoles HTTP / SOAP pour la réponse à envoyer au client.

Le "WSIG agent" est la passerelle entre le web et le monde des agents, et est responsable de :

- Transférer des actions reçues par le "WSIG servlet" aux agents qui sont en mesure de les exécuter, et puis collecter les réponses.
- S'enregistrement dans le DF du JADE afin de recevoir des notifications sur les enregistrements des agents dans le système.
- Créer le fichier WSDL correspondant à chaque service d'agent enregistré par le DF, et publier le service dans l'UDDI, si nécessaire.

## 4.2 Les composants du noyau fonctionnel (NF)

### 4.2.1 Les composants de la couche N-1

Dans cette section, nous présentons les diagrammes de classes des agents de communication, de coordination et de production, respectivement, et qui représentent l'environnement des agents dans l'$U3D$. Ces classes peuvent être interfacées avec l'$UDDI4C$, mais pas obligatoirement. Les classes d'agents `AgentCommunication`, `AgentCoordination` et `AgentProduction` possèdent des classes internes qui héritent de divers comportements et protocoles d'interaction offerts par JADE : `CyclicBehaviour`, `AchieveREInitiator` / `AchieveREResponder` et `ContractNetInitiator` / `ContractNetResponder`.

#### 4.2.1.1 Diagrammes de classes des différents agents

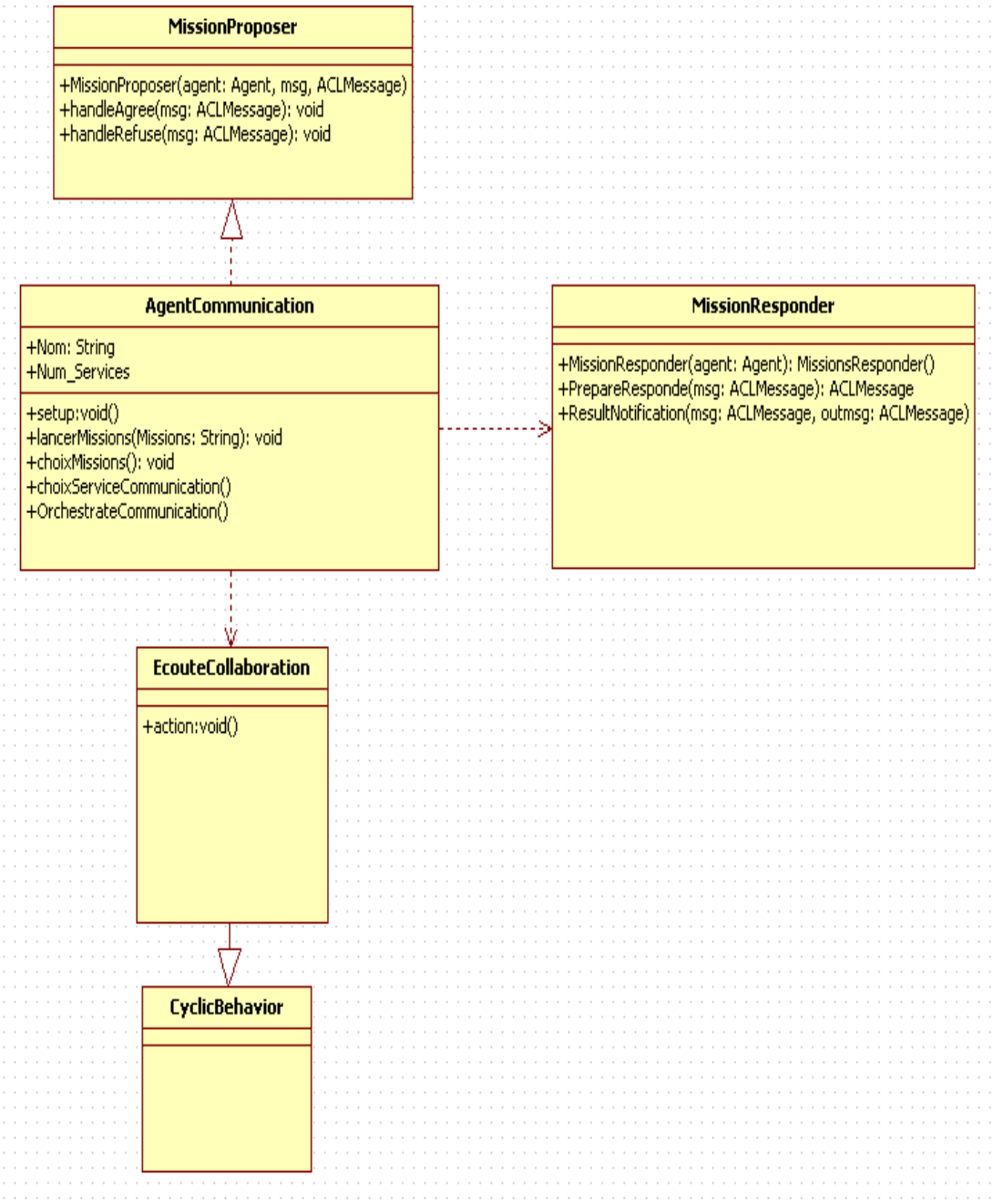



FIG. 4.2 – Diagramme de classe - Agent Communication

Dans la figure 4.2, nous illustrons le diagramme de classes des agents de communication. La classe `AgentCommunication` utilise, en plus des classes héritées de JADE, de diverses classes pour le choix de missions (principalement des missions d'intégration ou de composition).

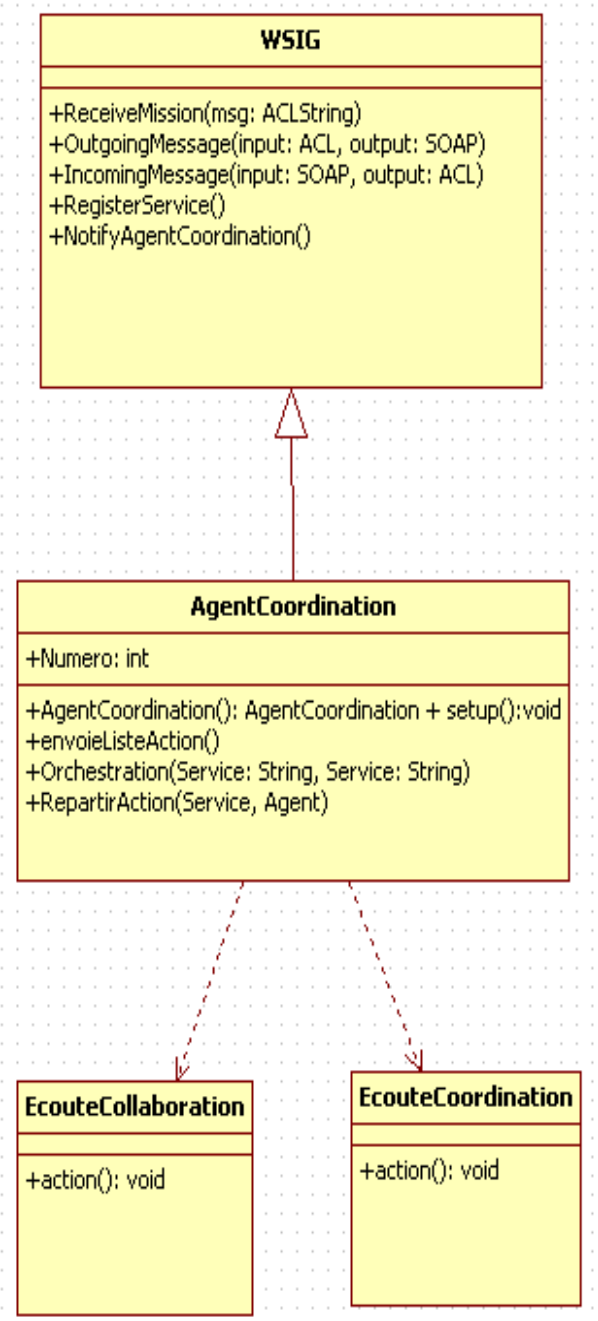



FIG. 4.3 – Diagramme de classe - Agent Coordination

Dans la figure 4.3, nous illustrons le diagramme de classes des agents de coordination. La classe `AgentCoordination` utilise de diverses classes pour l'écoute de la collaboration et de la coordination.

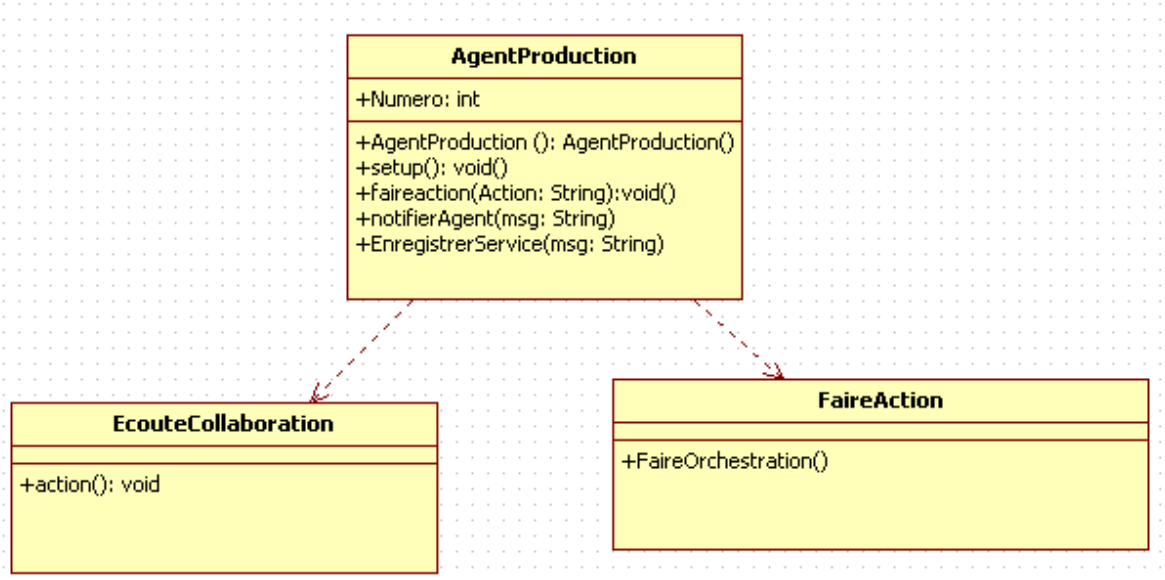



FIG. 4.4 – Diagramme de classe - Agent Production

Dans la figure 4.4, nous illustrons le diagramme de classes des agents de production. La classe `AgentProduction` utilise de diverses classes pour exécuter les actions issues de la phase de coordination.

### 4.2.2 Les composants de la couche N du NF

Le NF sur le niveau N permet des mécanismes pour traduire des invocations de services web à un langage compris par les agents logiciels, et vice versa. Notre objectif principal est de se disposer d'un outil qui traduit ces invocations, et de créer des mécanismes d'interactions afin d'assurer la malléabilité du système.

#### 4.2.2.1 Le Web Service Integration Gateway (WSIG)

Comme le montre la figure 4.5, le NF au niveau N contient un composant WSIG ainsi que plusieurs codecs, chacun lié, directement ou indirectement, à deux registres :

- Le Directory Facilitator (DF) utilisé par la plateforme JADE pour enregistrer les agents, et qui n'est pas visible de l'extérieur de la plateforme.
- L'UDDI qui est visible en interne par le WSIG, et de l'extérieur pour des services web sur internet, mais pas directement aux agents. Ainsi, le WSIG agit comme l'interface entre les agents et l'environnement SOA.

Afin d'être visible dans les deux environnements, le WSIG est enregistré comme un agent dans le DF et un service web dans l'annuaire UDDI, où toute description de service enregistré dans le DF ou l'UDDI est automatiquement traduite en une entrée pour l'autre. Le but est de veiller à ce que tout service web enregistré est visible à des agents par l'intermédiaire du DF, et tout agent est visible pour les clients des services web via l'UDDI.

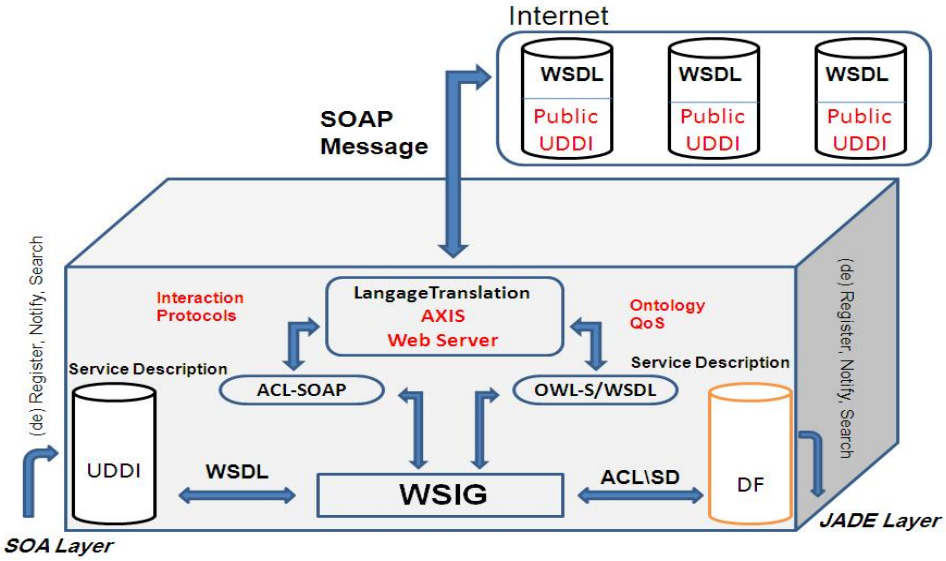



FIG. 4.5 – Les composants de la couche N de l'architecture

Nous décrivons les composants locaux du WSIG :

- Le OWL-S/WSDL codec qui a comme but de générer des ontologies OWL-S à partir des spécifications de WSDL. Un besoin nécessaire est d'avoir un composant qui fait la conversion des opérations trouvant dans le fichier WSDL, à des procédures atomiques de OWL-S. Evidemment, pas toutes les opérations dans le fichier WSDL peuvent être converties à des descriptions ontologiques. Ainsi, ces descriptions vont être rajoutées manuellement par le programmeur.

- Le codec ACL-SOAP est responsable de l'analyse des messages ACL provenant des agents. Il les traduit à des invocations de services web, et finalement restitue les résultats pour l'enregistrement dans l'UDDI. Ce codec fonctionne aussi dans une manière bidirectionnelle afin de traduire des messages SOAP à des spécifications de messages ACL correctement encodés.

- Le serveur Axis' JAX-RPC (Java API for XML Remote Procedure Call) est une interface de programme d'application (API) qui permet aux développeurs Java d'inclure des appels de procédures à distance (RPC) avec n'importe quelle application web. En effet, Il permet aux applications ou aux services web d'appeler plus facilement d'autres applications. Le JAX-RPC simplifie le développement en faisant une abstraction des mécanismes de SOAP en cours de l'exécution, en fournissant des mécanismes de conversion entre le langage Java et la description des services web en WSDL.

L'objectif principal est d'assurer la cohérence du système en liant un agent à son service correspondant, par le biais du WSIG. Par exemple, un service web de communication est géré par un agent de communication, assurant son intégration et son invocation dans l'interface physique du système. Ainsi, le WSIG expose les services fournis par les agents, et les publies dans le DF comme étant des services web. Le processus implique la création d'un fichier WSDL pour chaque description de service enregistré dans le DF, et éventuellement la publication des services exposés dans le répertoire UDDI. Dans les

figures 4.6 et 4.7, nous pouvons voir des captures d'écrans de l'interface du WSIG, avec les services enregistrés et leurs opérations, ainsi que les agents qui gèrent ces services.

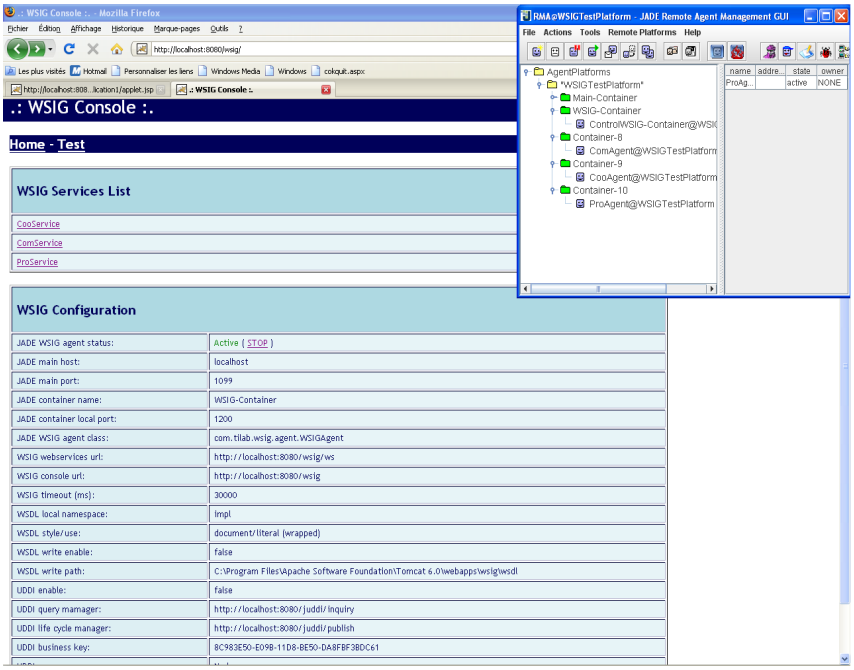



FIG. 4.6 – Capture d'écran de l'interface WSIG du système

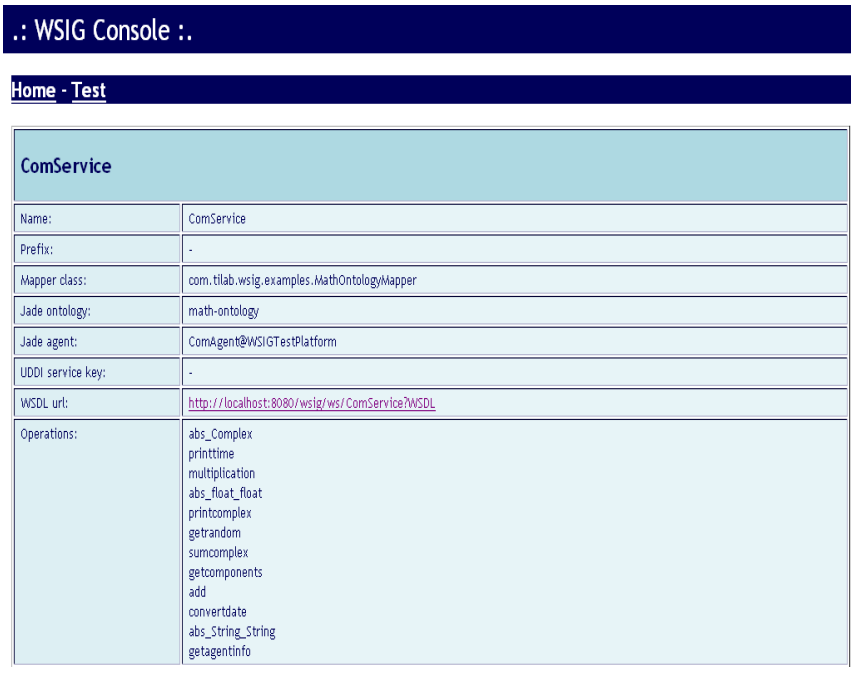



FIG. 4.7 – Un exemple d'un service de communication enregistré dans le WSIG

### 4.2.2.2 Les opérations de WSIG

Le fonctionnement de ce composant est relativement transparent, avec l'administrateur qui a pour tâche principale de coder les comportements des agents, qui sont nécessaires

pour interagir avec les services web, et comprendre leurs protocoles pour pouvoir les invoquer. Cette information est présente dans la description de DF du service web concerné. Un point important est que toutes les interactions entre le WSIG et les agents utilisent des messages FIPA-ACL "Request" et FIPA-ACL "Inform", ce qui simplifie le fonctionnement du WSIG, et est nécessaire pour invoquer des messages "request-response" des services web. Dans notre système, le WSIG aura le rôle d'un "proxy" entre les agents et les services web. En d'autres termes, ce composant reçoit les invocations des services web par les agents, recherche le service sur internet en construisant une requête SOAP à partir des messages ACL provenant des agents. Le service web trouvé est ensuite enregistré dans le système selon les fonctionnalités qu'il offre (communication, coordination ou production).

Comme mentionné, le WSIG utilise le DF et l'UDDI pour gérer les services et les agents. Les actions typiques associées à ces deux répertoires sont "enregistrer", "supprimer", "modifier" et "rechercher". Dans le cas du répertoire DF qui a une visibilité limitée, ces actions sont à la disposition des agents JADE seulement. Pour l'UDDI, ces actions sont visibles aux services web externes et leurs clients. La différence avec ce dernier est qu'il est également visible au WSIG, mais indirectement à tout autre agent JADE. Toutes les actions d'enregistrement, de suppression et de modification effectuées dans la plateforme sont automatiquement reprises dans le répertoire homologue. Par exemple, si un service d'agent est supprimé du DF, le WSIG va faire de sorte que le "tModel", ou la référence correspondante, soit retirée de l'UDDI. Cela permet de s'assurer que les deux répertoires restent cohérents.

## 4.3 Les processus de malléabilité dans le NF (Interactions entre les couches N et N-1)

### 4.3.1 Invocation d'un service web par un agent JADE

Le WSIG est mis en œuvre dans notre système afin d'assurer la malléabilité par des mécanismes d'intégration et de composition. Ces mécanismes sont mis en œuvre par des protocoles d'orchestration, où le WSIG agit comme un "chef d'orchestre" afin d'assurer la génération de nouveaux comportements à partir de ces mécanismes. Dans la figure 4.8, nous présentons le processus d'intégration d'un service externe. Quelques mots clés utilisés :

- `SD` : "Service Description" ou la description du service.
- `ACL` : "Agent Communication Language", ou le langage de communication utilisé par les agents.
- `DF` : "Directory Facilitator", ou l'annuaire des agents.
- `tModel` : Une entrée ou un identifiant du service web.
- `Stub` : Lien entre le service et le système qui l'invoque.

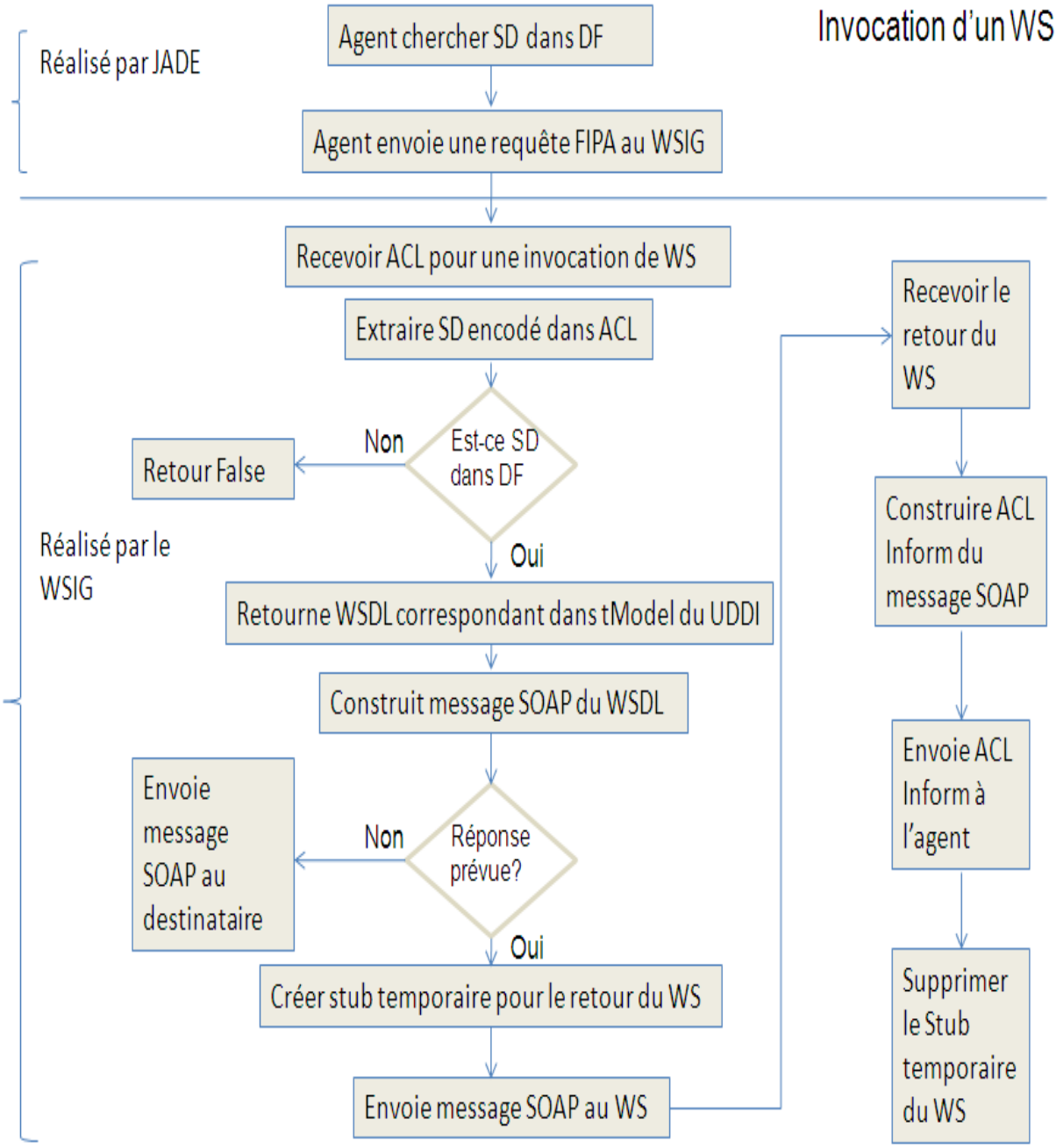



FIG. 4.8 – Invocation d'un service web par un agent JADE

### 4.3.2 Composition de deux services internes

Le deuxième processus est la composition de deux ou plusieurs services. Ce processus est mis en œuvre avec un protocole d'orchestration par le WSIG, qui assure que les services à composer vont générer un nouveau comportent. Le WSIG reçoit la requête pour composer deux services enregistrés dans le système. Ainsi, il extrait les informations nécessaires, notamment les informations sémantiques contenues dans l'ontologie qui décrit leurs comportements. A partir de ces informations, le WSIG teste si la composition aboutit à un nouveau comportement. Nous pouvons voir ce processus dans la figure 4.9, où le WSIG teste les "input" et "output" (IO1, IO2) pour savoir s'ils génèrent un nouveau effet

(E) dans le système. Si oui, le WSIG crée une nouvelle entrée (”tModel”) et avertie l'agent qui a envoyé la requête initiale pour la création du service.

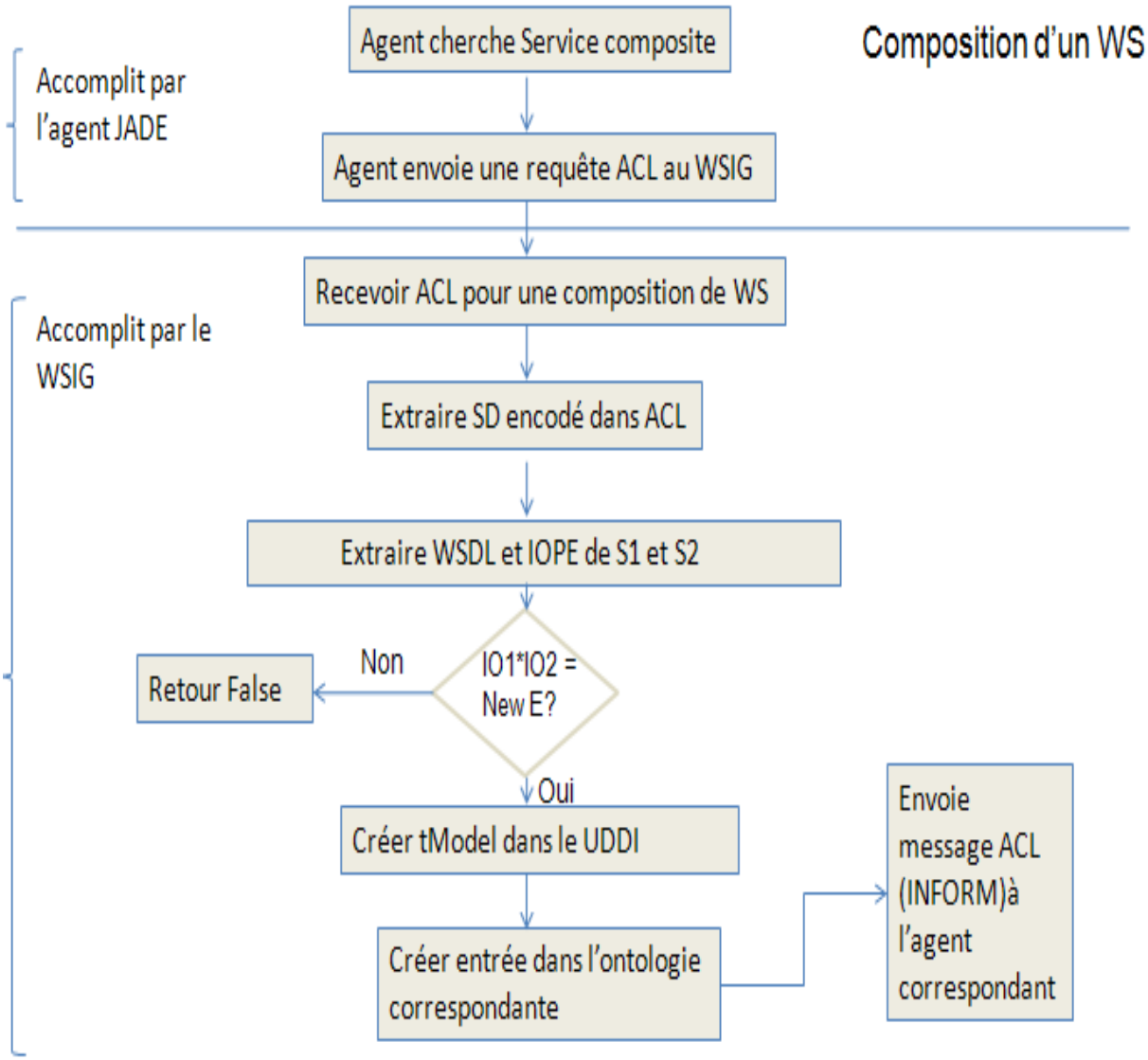



FIG. 4.9 – Composition de deux services

### 4.3.3 Enregistrement d'un service web comme service d'agent

Le WSIG permet à un service web externe de publier une description de service comme étant un service d'agent dans la plateforme. En effet, la description WSDL du service web est enregistrée auprès de l'UDDI connecté avec le WSIG. Quand un nouvel enregistrement est détecté, le WSIG le traduit en une entrée équivalente, et l'envoie pour s'enregistrer dans le répertoire DF. En raison de la différence évidente entre les descriptions WSDL et ACL, certains attributs de l'entrée dans l'UDDI seront traduits et insérés dans des propriétés d'attributs de l'agent dans le DF. Nous pouvons voir le processus dans la figure 4.10.

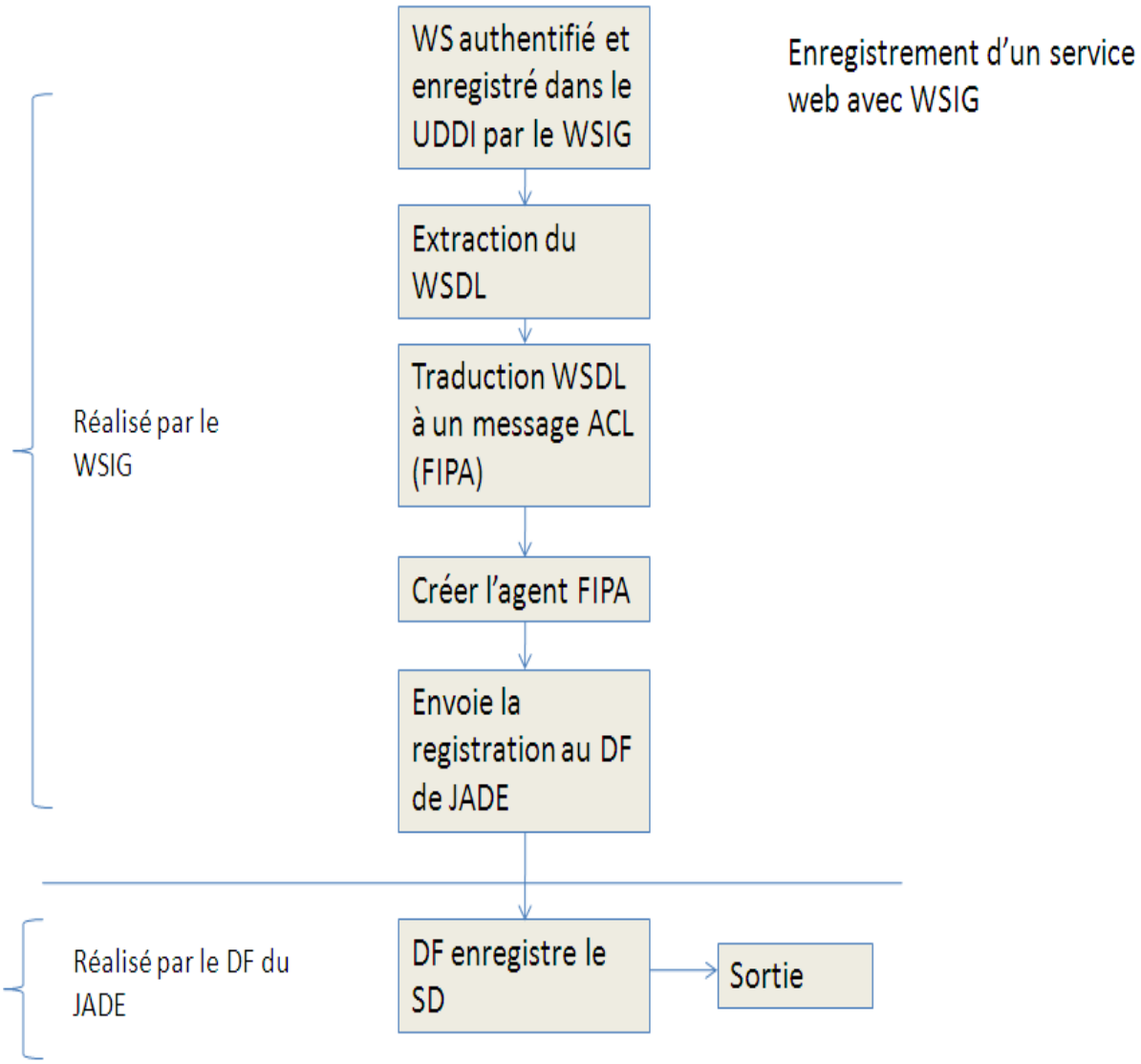



FIG. 4.10 – Enregistrement d'un service web par un agent

### 4.3.4 Diagrammes de séquences et algorithmes pour l'intégration et la composition

Dans cette section, nous présentons des diagrammes de séquences ainsi que des algorithmes afin de modéliser les interactions qui se présentent dans le système. Nous rapellons que notre travail consiste à mettre en œuvre une collaboration machine-machine. Ainsi, la machine est au centre de la collaboration et non l'utilisateur.

#### 4.3.4.1 Diagrame de Séquence

Dans la figure 4.11, nous présentons le diagramme de séquence des évènements et des interactions produites par le WSIG. Dans les figures 4.12 et 4.13, nous trouvons les diagrammes de séquences des deux mécanismes d'intégration et de composition de services.

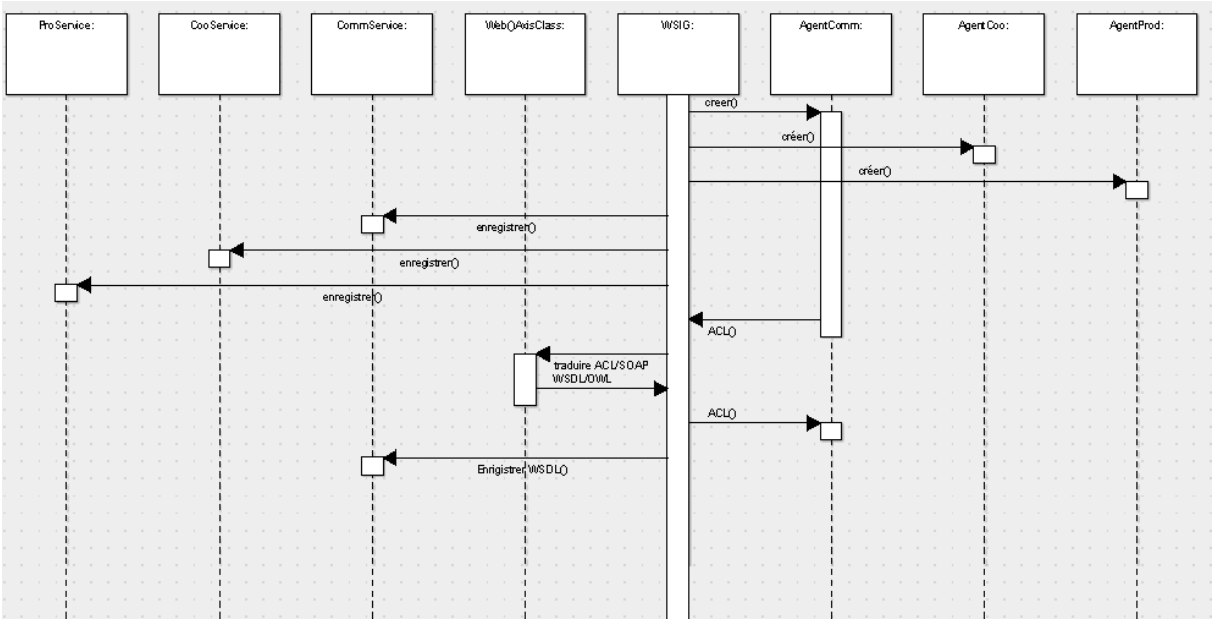



FIG. 4.11 – Diagrame de séquence - WSIG

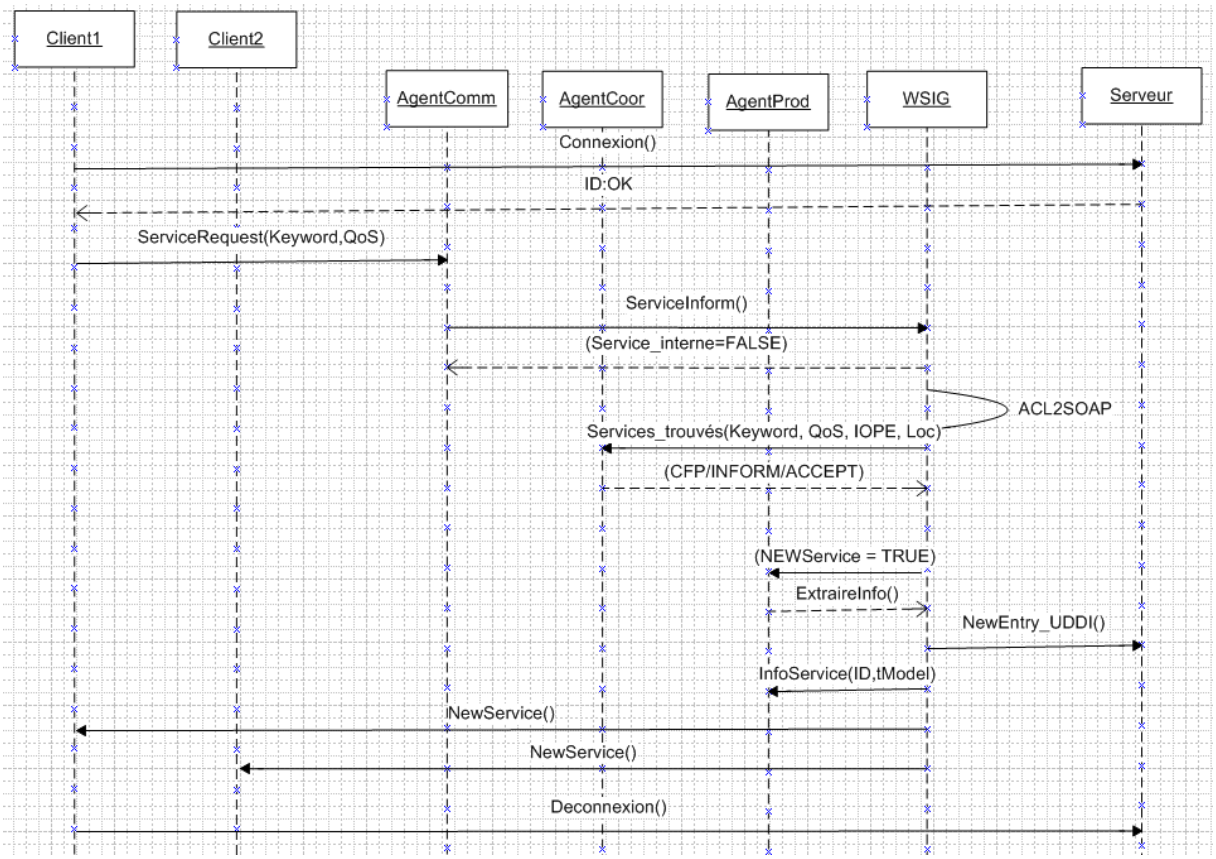



FIG. 4.12 – Diagramme de séquence pour l'intégration d'un service

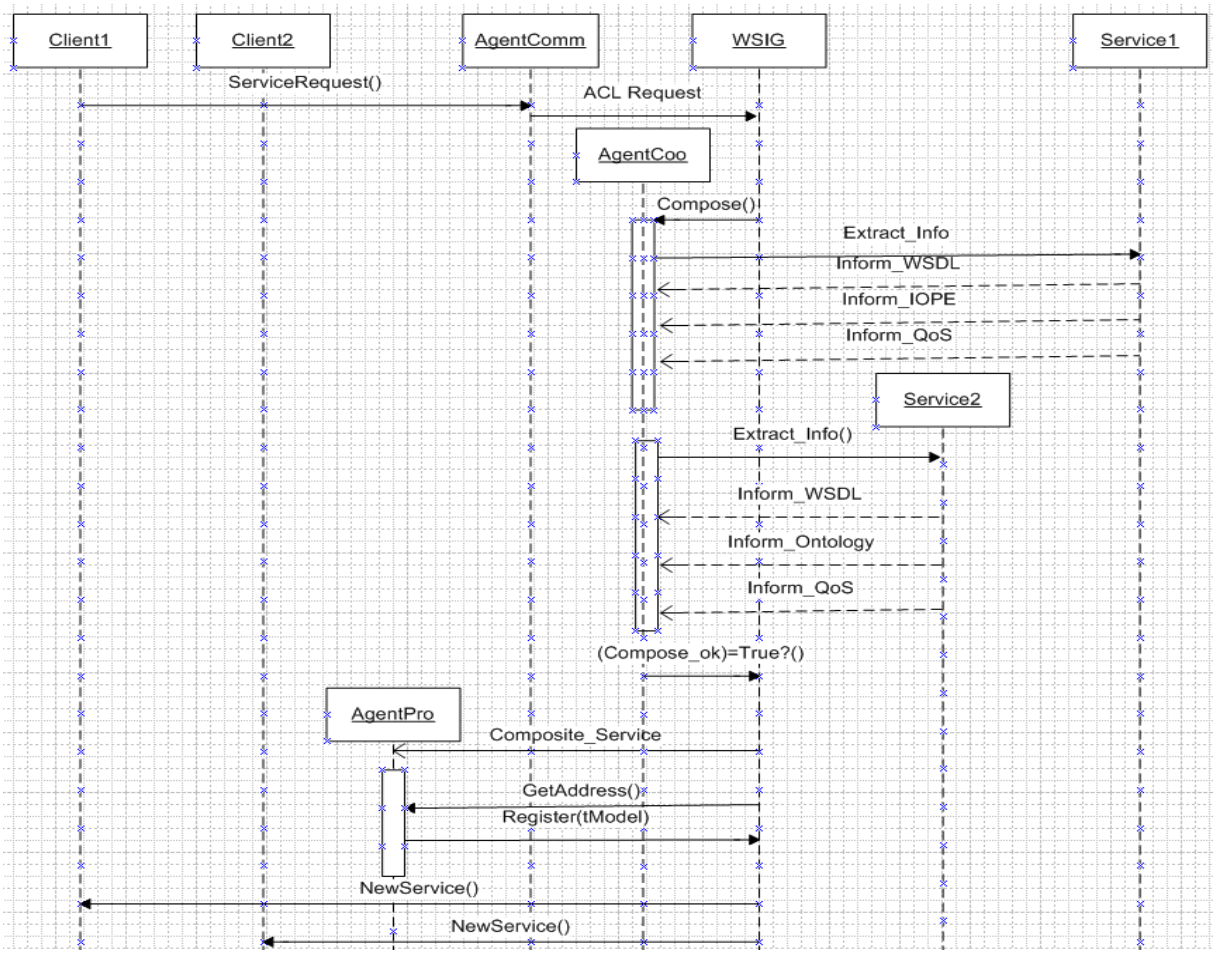



FIG. 4.13 – Diagramme de séquence pour la composition d'un nouveau service

### 4.3.4.2 Algorithmes de malléabilité

Dans cette section, nous présentons des algorithmes qui prennent en considération les rôles des trois agents de communication, de coordination et de production dans les processus de l'intégration et de composition de services. Ces algorithmes sont représentés dans les figures 4.14, 4.15 et 4.16, respectivement, et s'appliquent aux deux architectures logicielles présentées dans le chapitre précédent : l'$UDDI4C$ et l'$U3D$. Ensuite, nous présentons l'algorithme de traduction de messages entre les services et les agents, afin d'interagir avec un service web externe (Figure 4.17). Ces algorithmes mettent en œuvre le rôle du WSIG afin d'orchestrer les tâches entre les agents du système pour définir les mécanismes pour la malléabilité.

Dans la phase de communication (Figure 4.14), et dans le cas d'une composition (Ligne 3), le WSIG est avertie de la mission et extrait les informations non-fonctionnels des deux services $i$ et $j$ (Ligne 4 et 5). Après, le WSIG envoie ces informations aux agents de coordination (Ligne 7). Dans le cas d'une intégration de services, le WSIG est notifié du service à intégrer ainsi que de sa qualité de service QoS (Ligne 8,9 et 10). Les actions à accomplir sont ainsi envoyés aux agents de coordination (Ligne 12 et 13).

```
1. Communication ( Comm_i) :
2.   tant que (MessageQueue NON vide  Service NULL) faire
3.     si (Mission_i = COMPOSE) alors
4.     WSIG ⟵ missionCOMPOSE(i,j) ;
5.     WSIG ⟵ QoS(i,j) ;
6.     si (QoS = TRUE) alors
7.       send CooAgent ⟵ (ComService_i,AgentComm_i,WSIGi, TRUE) ;
8.     si (Mission_i = INTEGRATE) alors
9.     WSIG ⟵ missionINTEGRATE(i) ;
10.    WSIG ⟵ QoS(i) ;
11.      send (Actions_i, WSIGi) ;
12.    si (QoS & Loc = TRUE) alors
13.      send (ComService_i,AgentComm_i,WSIGi, TRUE) ;
```

FIG. 4.14 – L'algorithme de communication d'un agent $Comm_i$ avec un service

Dans la phase de coordination (Figure 4.15), et après la phase de communication (Ligne 2), les agents de coordination reçoivent le message contenant les descriptions des services (Ligne 3, 4 et 5). Dans le cas d'une composition (Ligne 2), les agents de coordination extraient et testent les IOPEs des deux services à composer, pour veiller à ce que le produit génère un service avec un nouveau comportement (Ligne 8 et 9). Ensuite, ils envoient les résultats au WSIG (Ligne 10). Dans le cas d'une intégration (Ligne 11), les agents de coordination reçoivent les informations non-fonctionnelles relatives aux services à intégrer, puis teste son QoS ainsi que sa localisation (Ligne 13 à 16). Si sa QoS est satisfaisante, le service est extrait et envoyé au WSIG (Ligne 17 à 19).

```
1. Coordination ( Coor_i) :
2.     si (Communication = TRUE) alors
3.     AgentCoo ⟵ receiveMessage(WSIG) ;
4.       AgentCoor_j ⟵ message.getContent(1) ;
5.       Mission_i ⟵ message.getContent(2) ;
6.     si (Mission_i = COMPOSE) alors
7.       Get(Source_i, Source_j) ⟵ (ServiceIOPEi.ServiceIOPEj) ;
8.       si((Inf(IOPE, Loc)Source_i) * (Inf(IOPE, Loc)Source_j)) == TRUE) ;
9.       Actions_i ⟵ ChoiceActionsWith(AgentCoor_j, Mission_i) ;
10.      send (Actions_i, WSIGi) ;
11.    si (Mission_i = INTEGRATE) alors
12.    AgentCoo ⟵ receiveMessage(WSIG) ;
13.      AgentCoor_j ⟵ message.getContent(1) ;
14.      Mission_i ⟵ message.getContent(2) ;
15.    AgentCoo ⟵ testQoS() ;
16.    AgentCooo ⟵ testLoc() ;
17.    si (QoS & Loc = TRUE) alors
18.      Service ⟵ message.getContent(WSIG) ;
19.      send (Service_i,AgentCoo_i,WSIGi, TRUE) ;
```

FIG. 4.15 – L'algorithme de coordination d'un agent $Coor_i$ et un Service $Comm_j$

Après la phase de communication et de coordination, la phase de la production (4.16) commence, qui contient les résultats de la collaboration. Ainsi, Les agents de production reçoivent le message (Ligne 2 et 3). Si un service est composé ou intégré, l'agent de production l'enregistre dans le système avec une nouvelle entrée dans l'UDDI. Les résultats sont envoyés au WSIG, et la phase de production est terminée (Ligne 4 à 8).

1. **Production (** $Prod_i$**) :**
2. si (Coordination = TRUE) alors
3. AgentPro ⟵ receiveMessage() ;
4. si (message.compose = TRUE OR message.integrate=TRUE) alors
5. Register NewServicePro ⟵ AgentPro() ;
6. $Results_i$ ⟵ ExecuteActions($Actions_i$) ;
7. send ($Results_i$, $WSIG_i$) ;
8. Terminate ($Prod_i$, $WSIG_i$) ;

FIG. 4.16 – L'algorithme de production d'un agent $Prod_i$ et un service $Comm_j$

Dans l'algorithme ci-dessous, nous présentons les messages envoyés par le système pour interagir avec un service externe, dans le cas d'une intégration. Ainsi, les agents de communication reçoivent les informations relatives à ces services, qui sont décris par les utilisateurs, et les envoient au WSIG (Ligne 4 à 7). Le WSIG crée une invocation `SOAP` qui contient les informations relatives au service, notamment à partir de son fichier WSDL et sa description syntaxique (Ligne 8). Le WSIG reçoit le message `SOAP` entrant avec les informations relatives au service. Ainsi, il les envoie au UDDI local (Ligne 10 et 11), et avertit l'agent responsable de la requête de l'accomplissement de la tâche (Ligne 13, 14 et 15).

1. **Translate (**$Comm_i$**,** $Coo_i$**,** $Prod_i$**) :**
2. tant que (MessageQueue NON vide) faire
3. message ⟵ receiveMessage() ;
4. si (message.sender = $Comm_j$) alors
5. $AdressComm_i$ ⟵ message.getContent(1) ;
6. $ACLComm_i$ ⟵ message.getContent(2) ;
7. send $WSIGi(AdressComm_j, ACLComm_i)$ ;
8. $SOAPOUT_j$ ⟵ ($AdressComm_j$,$ACLComm_i$) ;
9. si ($SOAPOUT_j(Comm_i) = TRUE$) alors
8. $SOAPIN_j$ ⟵ ($SOAPOUT_j$,$AdressCommS_j$, $QoS_i$) ;
10. $WSIGj$ ⟵ ($SOAPIN_j$) ;
11. send $UDDI_k$ ($SOAPIN_j$) ;
13. Translate $ACL_j$ ⟵ ($SOAPIN_j$) ;
14. send $DF_k$ ($ACL_j$, $Agent_j$) ;
15. send ($Results_i$, $Coll_j$) ⟵ $AgentInterfacej$ ;

FIG. 4.17 – Algorithme de consctruction d'une invocation de services web

Dans cette section, nous avons présenté la mise en œuvre et l'implémentation de notre architecture collaborative $U3D$. Dans la section suivante, nous mettons en œuvre des

protocoles d'orchestration afin de mesurer la performance des deux types d'architectures logicielles présentées, $UDDI4C$ et $U3D$. A partir de ces deux études, nous comparons les résultats avec le formalisme SMA-C du modèle C4 (Khezami, 2005). Le but est de calculer le coût de l'application de la malléabilité dans les systèmes collaboratifs à bases de services web et/ou agent logiciels.

## 4.4 Etude et comparaison de performance

Nous reprenons l'étude faite pour mesurer la performance de SMA-C (Khezami, 2005) (le formalisme associé au modèle C4 présenté dans le chapitre 1), en termes de nombres de messages émis. Ainsi, nous identifions le nombre de messages que le système émet dans chaque phase de la collaboration. Après, nous appliquons l'architecture $UDDI4C$ comme un support à la malléabilité pour le SMA-C, qui est initialement conçu pour l'assistance au télétravail collaboratif, appliqué à la téléopération d'un robot. Nous déduisons le nombre de messages émis suite à l'application de la nouvelle architecture, pour mesurer le coût de la malléabilité sur la performance du système. Ainsi, nous ajoutons le nombre de messages obtenues suite à l'application du l'$UDDI4C$ sur les messages initiaux du formalisme SMA-C.

Ensuite, nous analysons l'architecture $U3D$, comme étant une architecture générique dédiée à la malléabilité des collecticiels. Dans cette architecture, les agents logiciels ont le seul objectif d'appliquer le mécanisme d'orchestration (intégration ou composition), afin de mettre en œuvre la malléabilité. Nous comparons les résultats avec ceux du SMA-C en termes de nombre de messages émis. Cette étude de performance est effectuée sans l'implication des utilisateurs dans les deux architectures. Nous commençons par étudier le formalisme SMA-C, en termes de nombre de messages émis, avant l'application de l'$UDDI4C$ comme un support à la malléabilité.

### 4.4.1 Etude des interactions dans l'architecture C4

#### 4.4.1.1 Communication

Dans cette phase, les agents de coordination et de production n'émettent pas de messages. Ainsi, le nombre de messages est calculé comme suit :

- $NbAgents - 1$ messages `REQUEST` envoyés par un agent (maître) à ses futurs collègues pour constituer un groupe
- $NbAgents - 1$ réponses
- $3 * NbAgents$ messages pour les agents de communication / coordination / production pour l'envoi du numéro de groupe d'agents à créer
- $NbAgents$ messages pour les agents de communication pour l'envoi de la liste des missions à effectuer

$$NbMComm = 6 * NbAgents - 2 \tag{4.1}$$

Lors de la simulation effectué pour mesurer le nombre de messages dans cette phase, la part de la communication peut augmenter ou se réduire d'un certain nombre de messages : comme le choix des missions est basé sur un vote des agents, celui-ci est aléatoire et peut

être reconduit plusieurs fois, augmentant par là-même le nombre de messages envoyés par les agents de communication.

#### 4.4.1.2 Coordination

Dans cette phase, deux types d'agents interviennent : les agents de collaboration, qui s'envoient le choix de mission en début de coordination, et les agents de coordination qui se répartissent les actions.

Ceci s'explique par les messages suivants :
- $NbActions * (NbAgents - 1) * 3$ messages pour les enchères liées à chaque action : un `CFP`, un `PROPOSE` et un `REJECT-PROPOSAL` ou `ACCEPT-PROPOSAL` pour chaque agent du groupe.
- $NbActions$ messages émis par les agents de coordination remportant les enchères vers leurs agents de collaboration.

Comme le modèle C4 est considéré comme "rigide", le nombre d'actions à effectuer est fixé manuellement à 8 actions. Ainsi, pour un nombre d'actions de 8, nous trouvons :

$$NbMCoor = 24 * NbAgents - 16 \tag{4.2}$$

#### 4.4.1.3 Production

Nous constatons ici que certains nombres de messages ne dépendent pas du nombre d'agents : les agents de coordination comme de production envoient un nombre de messages égal au nombre d'actions. Il s'agit respectivement des $NbActions$ messages émis par l'agent de coordination maître pour signaler qu'une action doit être faite et des $NbActions$ messages émis par la production après accomplissement d'une action. Ces messages sont répartis de la façon suivante :

- $NbActions * (NbAgents - 1)$ messages émis par l'agent de collaboration maître pour signaler le début d'une action.
- $NbActions * NbAgents$ messages envoyés par l'agent de collaboration à son agent de production.
- $NbActions * NbAgents$ messages de fin d'action.
- $NbActions$ messages envoyés par l'agent de collaboration maître à son agent de coordination pour signaler une fin d'action.
- $NbAgents * 3$ messages pour mettre fin aux agents de communication, de coordination et de production en fin de processus.
- $NbAgents - 1$ messages envoyés par l'agent de collaboration maître pour tuer les autres agents collaboration.

Comme le nombre d'actions est fixé à 8, le nombre totale de messages dans cette phase est :

$$NbMColl = 28 * NbAgents - 9 \tag{4.3}$$

Dans ce qui suit, nous étudions les interactions dans les deux architectures logicielles proposées :

– L'architecture $UDDI4C$, qui agit comme un support pour la malléabilité pour les systèmes collaboratifs à bases de multi-agents ou de services web. Le but de cette architecture est d'assister ces systèmes pour introduire une certaine malléabilité. Ainsi, nous appliquons l'architecture $UDDI4C$ sur le modèle SMA-C, afin de mettre en œuvre la malléabilité par l'intégration/composition de services.

– L'architecture $U3D$, qui représente une architecture complète et générique à base de services web et d'agents logiciels. Ainsi, cette architecture logicielle n'assiste pas les systèmes existants. Il s'agit plutôt d'une nouvelle architecture logicielle pour introduire la malléabilité dans la conception des systèmes collaboratifs génériques. Dans cette architecture logicielle, le mécanisme d'orchestration proposé décrit un chef d'orchestre qui sera chargé de mettre en œuvre la malléabilité dans le système.

## 4.5 Etude des interactions dans l'architecture $UDDI4C$

Nous reprenons l'étude du formalisme SMA-C, en rajoutant le nombre de messages nécessaires pour soutenir la malléabilité en l'interfaçant avec l'$UDDI4C$. Dans ce qui suit, nous étudions les interactions du système en appliquant le processus d'intégration et de composition, respectivement.

### 4.5.1 Intégration de services

#### 4.5.1.1 Communication

A partir du formalisme décrit dans la section (3.2.1.1), le nombre de messages pour le support de la malléabilité dans cette phase est calculé comme suit :

– $NbInf_i$ messages `REQUEST` envoyés par les agents de communication pour intégrer un service $i$
– $NbInf_i$ messages `INFORM` envoyés par l'agent de collaboration aux agent de communication pour la recherche du service localement comme un étant un service d'agent (nous supposons que le service n'est pas présent dans le système).
– $NbInf_i$ messages envoyés par l'agent de collaboration aux agents de communication signalant la recherche du service à partir de sa description syntaxique (WSDL). Un message SOAP est créé à partir de la description syntaxique du service.

Ainsi, le nouveau nombre total de messages dans le SMA-C après l'application de l'$UDDI4C$ est calculé comme suit :

$$NbMComm = (6 * NbAgents - 2) + 3 * \texttt{NbAgents} \tag{4.4}$$

Où `3* NbAgents` représente le coût de la malléabilité pour la phase de la communication.

#### 4.5.1.2 Coordination

Dans cette phase, les messages échangés sont les suivants :

– $NbActions_i$ messages `INFORM` envoyés par l'agent de collaboration aux agents de coordination, pour leur informer des services trouvés ainsi que leurs qualités de services (QoS).
- $NbAction_i$ messages `REQUEST` envoyés par l'agent de collaboration pour selectionner le service qui répond aux besoins non-fonctionnels des agents de coordination.
- $NbAction_i$ messages `REQUEST` envoyés par l'agent de collaboration pour invoquer le service qui correspond à la requête initiale, en examinant sa QoS et la description syntaxique des opérations qu'il contient.
- $NbAction_i$ messages `INFORM` émis par l'agent de collaboration aux agents de coordination pour signaler que le service est trouvé (ou non).

Pour un nombre d'actions de 8, nous trouvons :

$$NbMCoor = 24 * NbAgents - 16 + 4 * \texttt{NbAgents} \tag{4.5}$$

Où `4* NbAgents` représente le coût de la malléabilité pour phase de la coordination.

#### 4.5.1.3 Production

Les messages dans cette phase sont répartis de la façon suivante :
- $NbRésultats$ messages émis par l'agent de collaboration aux agents de production pour signaler l'intégration d'un nouveau service.
- $NbRésultats$ messages émis par les agents de production pour enregistrer le service $i$ dans le UDDI locale.

Du point de vue de la collaboration, nous obtenons :

$$NbMProd = 28 * NbAgents - 9 + 2 * \texttt{NbAgents} \tag{4.6}$$

Où `2* NbAgents` représente le coût de malléabilité pour la production.

### 4.5.2 Composition de services

#### 4.5.2.1 Communication

A partir du formalisme décrit dans la section (3.2.1.1), le nombre de messages pour le support de la malléabilité dans cette phase est calculé comme suit :
– $NbInf$ messages `REQUEST` envoyés par les agents de communication à l'agent de collaboration pour composer un service à partir de deux services locaux $i$ et $j$.
– $NbInf$ messages `REQUEST` envoyés par l'agent de collaboration aux services $i$ et $j$ pour extraire leurs informations sémantiques à partir de l'ontologie correspondante.

Ainsi, le nouveau nombre total de messages est calculé comme suit :

$$NbMComm = (6 * NbAgents - 2) + 2 * \texttt{NbAgents} \tag{4.7}$$

Où `2* NbAgents` représente le coût de la malléabilité pour la communication.

#### 4.5.2.2 Coordination

Dans cette phase, les messages échangés sont les suivants :

- *NbAction* messages `REQUEST` envoyés de l'agent de collaboration aux agents de coordination contenant les IOPEs des deux services $i$ et $j$ à composer.
- $NbAction_j$ messages `INFORM` émis par les agents de coordination, signalant à l'agent de collaboration si les services sont composables (ou non).

Ainsi, le nouveau nombre total de messages est calculé comme suit :

$$NbMCoor = 24 * NbAgents - 16 + 2 * \texttt{NbAgents} \tag{4.8}$$

Où `2* NbAgents` représente le coût de la malléabilité pour la coordination.

#### 4.5.2.3 Production

Les messages dans cette phase sont répartis de la façon suivante :
- *NbRésultats* messages émis par l'agent de collaboration aux agents de production pour signaler la composition d'un nouveau service.
- *NbRésultats* messages émis par les agents de production pour enregistrer le nouveau service dans le UDDI locale.

Du point de vue de la collaboration, nous obtenons :

$$NbMProd = 28 * NbAgents - 9 + 2 * \texttt{NbAgents} \tag{4.9}$$

où `2* NbAgents` représente le coût de malléabilité pour la production.

#### 4.5.2.4 Comparaison de performance

Dans le tableau ci-dessus, nous pouvons voir le nombre de messages émis dans le formalisme SMA-C, ainsi que les nouveaux messages en appliquant l'$UDDI4C$ sur ce dernier pour les deux mécanismes d'intégration ($UDDI4C_I$) et de composition ($UDDI4C_C$).

| Espace/Système | SMA-C | $UDDI4C_I$ | $UDDI4C_C$ |
|---|---|---|---|
| Communication | $6 * NbAgents - 2$ | $+3 * NbAgents$ | $+2 * NbAgents$ |
| Coordination | $24 * NbAgents - 16$ | $+4 * NbAgents$ | $+2 * NbAgents$ |
| Production | $28 * NbAgents - 9$ | $+2 * NbAgents$ | $+2 * NbAgents$ |

TAB. 4.1 – Comparaison des résultats de $SMA - C$ avec $UDDI4C_I$ et $UDDI4C_C$ dans les trois phases de la collaboration

Afin de calculer le coût de la malléabilité globale en appliquant l'$UDDI4C$ sur SMA-C, nous calculons le nombre de messages ajouté à ce dernier dans toutes les phases de la collaboration. Soit $x$ le nombre de messages émis, et $F(x)$ la fonction qui calcule le nombre de messages dans les trois phases de la collaboration :

$$F(x)_{collaboration} = F(x)_{communiation} + F(x)_{coordination} + F(x)_{production}$$

Le nombre initial de messages envoyés par le SMA-C est :

$F(x)_{collaboration} = 6 * NbAgents - 2 + 24 * NbAgents - 16 + 28 * NbAgents - 9$

$$\Rightarrow F(x)_{collaboration} = 58 * NbAgents - 27$$

Soit $F(x)_{collaboration_I}$ le nombre de messages émis dans SMA-C après l'application de l'architecture $UDDI4C$ pour le mécanisme de l'intégration. Le nombre total de messages ainsi émis est de :

$$F(x)_{collaboration_I} = 58*NbAgents-27+3*NbAgents+4*NbMAgents+2*NbMAgents$$

$$\Rightarrow F(x)_{collaboration_I} = 67 * NbAgents - 27$$

En ce qui concerne l'application de l'$UDDI4C$ pour la composition de services, nous retrouvons :

$$F(x)_{collaboration_C} = 58 * NbAgents - 27 + 2 * NbAgents + 2 * NbAgents + 2 * NbAgents$$

$$\Rightarrow F(x)_{collaboration_C} = 64 * NbAgents - 27$$

Pour calculer le coût de la malléabilité après l'application de l'$UDDI4C$, nous retrouvons pour le mécanisme de l'intégration :

$(F(x)_{collaboration_I} - F(x)_{collaboration})/F(x)_{collaboration} = 0.15$
$\Rightarrow$ **Le coût de malléabilité est de 15%.**

Pour calculer le coût de la malléabilité après l'application de l'$UDDI4C$, nous retrouvons pour le mécanisme de la composition :

$(F(x)_{collaboration_C} - F(x)_{collaboration})/F(x)_{collaboration} = 0.10$
$\Rightarrow$ **Le coût de malléabilité est de 10%.**

L'échantillon de test est composé d'un nombre d'agents variant de 2 à 35. Un extrait des données se trouvent dans les annexes A. Dans les figures 4.18, 4.19 et 4.20, nous pouvons voir la comparaison de performance avant et après l'application de l'$UDDI4C$ sur SMA-C pour chaque phase de la collaboration. Dans la figure 4.21, nous pouvons voir la comparaison des deux systèmes pour toutes les phases de la collaboration.

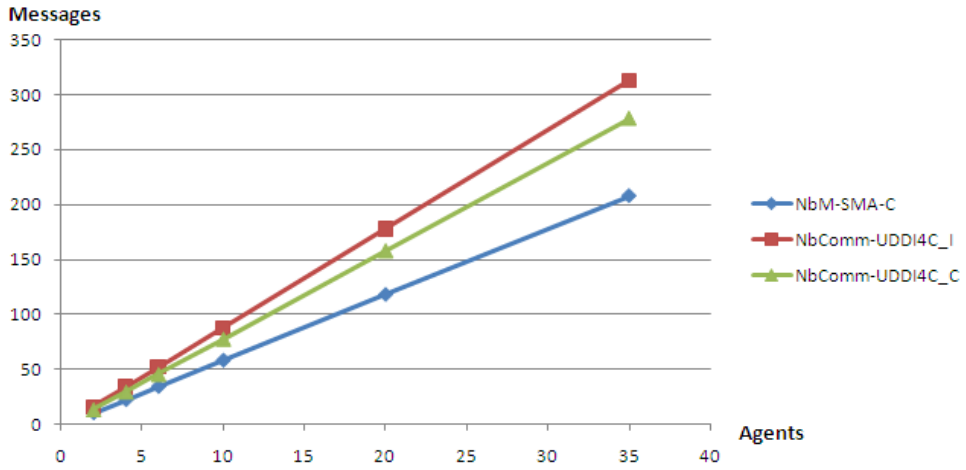



FIG. 4.18 – Comparaison de performance entre le SMA-C avant et après l'application de l'*UDDI4C* (pour l'intégration et la composition) dans la phase de communication

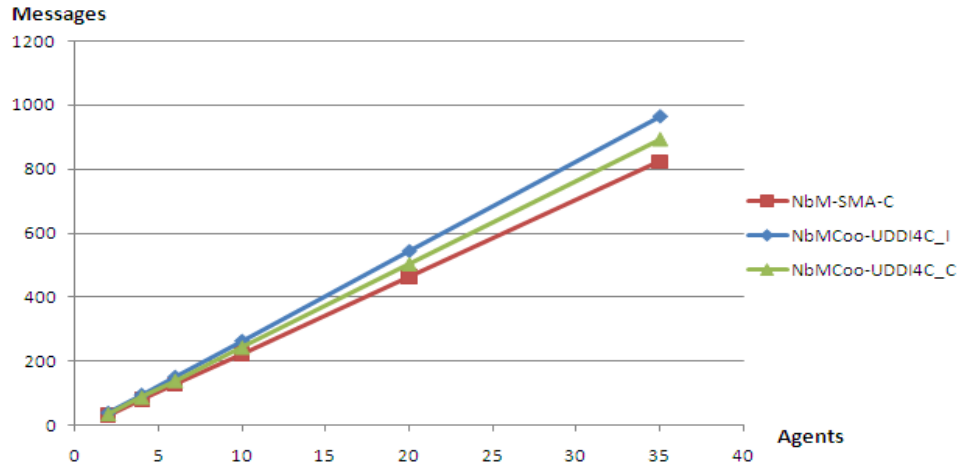



FIG. 4.19 – Comparaison de performance entre le SMA-C avant et après l'application de l'*UDDI4C* (pour l'intégration et la composition) dans la phase de coordination

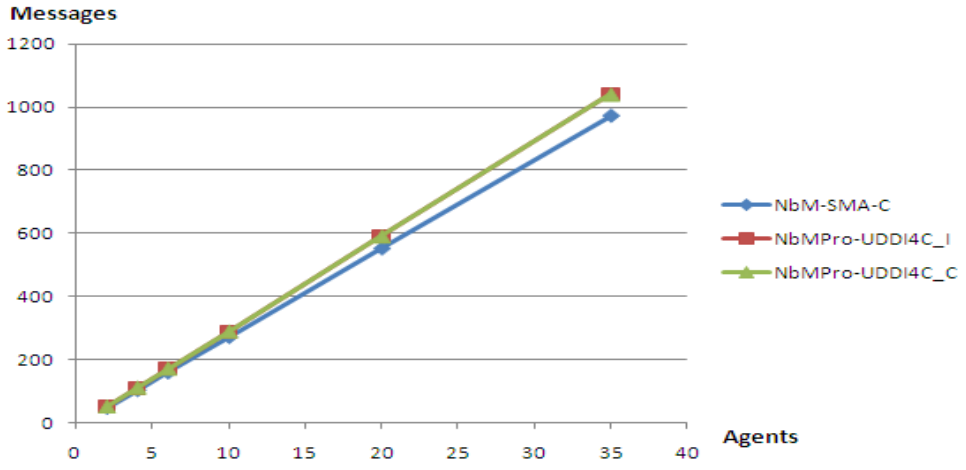



FIG. 4.20 – Comparaison de performance entre le SMA-C avant et après l'application de l'*UDDI4C* (pour l'intégration et la composition) dans la phase de production

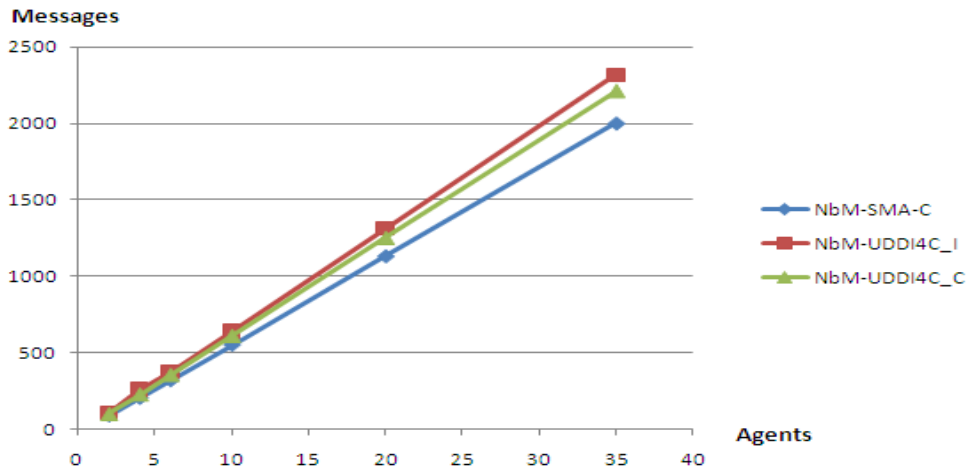



FIG. 4.21 – Comparaison de performance entre le SMA-C avant et après l'application de l'$UDDI4C$ (pour l'intégration et la composition) pour toutes les phases de la collaboration

## 4.6 Etude des interactions dans l'architecture $U3D$

Dans la section précédente, nous avons présenté une étude pour évaluer l'ancienne version du C4 en appliquant l'architecture logicielle $UDDI4C$. Ainsi, de nouvelles évaluations ont été faites pour mettre en œuvre la malléabilité des collecticiels. Notre but est de démontrer qu'avec la nouvelle architecture $U3D$, le système doit générer de meilleures performances au niveau de nombres de messages émis. Nous rappelons que nous voulons évaluer un système collaboratif malléable, où les agents communiquent et coordonnent afin de produire de nouveaux comportements. Afin de mettre en œuvre la malléabilité, deux mécanismes de collaboration se mettent en place :

- L'orchestration des services, où l'agent collaboration (Khezami, 2005) sera un agent WSIG (définit par un agent WAG dans l'architecture conceptuelle- chapitre 3), qui va mettre en œune collaboration malléable par l'intégration et la composition de services. En plus, les interactions entre les environnements des services web et celles des agents passent toujours par le WSIG, qui affecte des services aux agents de la même classe. Ainsi, il orchestre des processus d'interactions, ainsi que les traductions des messages entre les deux environnements.

- La chorégraphie, où les agents vont mettre en œuvre des scénarios potentiels d'interactions internes. Ainsi, l'agent WSIG n'aura pas d'influence sur les interactions générées (il perd son rôle de chef d'orchestre). Cette phase pourra être appliquée aux agents qui veulent se regrouper pour faire une classe d'agent (communication, coordination ou production). La chorégraphie repose sur chacun des services afin de répondre de façon appropriée, d'une part, à des erreurs ou des problèmes se produisent, et d'autre part pour composer de nouveaux classes de services. Dans ce processus, le WSIG n'agit en aucun cas sur le déroulement des interactions dans le système, qui sont prédéfinies afin de créer des classes de communication, coordination et production selon les fonctionnalités offertes par chacun des services.

L'étude du SMA-C est extraite d'un simulateur pour tester la performance du modèle C4 (Khezami, 2005). Ce simulateur est basé sur un envoi arbitraire de messages entre les agents de communication afin de dédier une action aux agents de coordination. Ceci implique un grand nombre de messages circulant dans le système, ce qui dégrade son performance. Notre but est de limiter le nombre de messages émis. Ainsi, en prédéfinissant les actions, le nombre de message sera plus précis, ainsi que l'action exécutée, qui constitue le produit de la collaboration et qui aura un impact sur la malléabilité du système. Dans notre étude, nous nous concentrons principalement sur le mécanisme d'orchestration. Ainsi, nous décrivons les processus d'intégration et de composition telle établie par ce mécanisme. En se basant sur les figures 4.24 et 4.26, nous déduisons le nombre de messages émis dans chaque phase de la collaboration.

## 4.6.1 Intégration de services

### 4.6.1.1 Communication

Les agents de coordination et de production n'émettent pas de messages pendant la communication. La communication est basée sur un échange d'information non-fonctionnels (principalement la QoS) entre, d'une part les agents de communication et l'agent de collaboration, et d'autre part entre l'agent de collaboration et le service déployé sur internet. Ainsi, le nombre de messages émis $NbMComm$ dans cette phase est le suivant :

- $NbMComm$ messages `REQUEST` envoyés par les agents de communication au WSIG afin de retrouver un service spécifique. Ces messages inclus les mots clés qui identifient le service, ainsi que des attributs non fonctionnels qui identifie sa qualité de service (QoS).
- $NbMComm$ messages `INFORM` envoyés par le WSIG aux agents de communication, indiquant la réception de la requête et le déclenchement du mécanisme de l'intégration.
- $NbMService$ messages `SOAP` crées, où les messages dépendent du nombre de services à intégrer (Figure 4.22). Pour la simplicité de l'étude, nous supposons qu'il s'agit d'un service à intégrer à chaque fois (nous considérons aussi que le service n'est pas retrouvé localement).

```
// Convert jade to soap
SOAPMessage soapResponse = null;
try {
    JadeToSoap jadeToSoap = new JadeToSoap();
    soapResponse = jadeToSoap.convert(operationAbsResult, wsigService, operationName);
} catch(Exception e) {
    log.error("Error in jade to soap conversion", e);
    throw new WSIGException(WSIGException.SERVER, e.getMessage());
}

// Send http response
try {
    sendHttpResponse(soapResponse, httpResponse);
} catch(Exception e) {
    log.error("Error sending http response", e);
    throw new WSIGException(WSIGException.SERVER, "Error sending http response. "+e.getMessage());
}

log.info("WSIG SOAP response sended, stop elaboration.");
```

FIG. 4.22 – Création d'un message SOAP à partir d'une requête d'un (ou plusieurs) agent(s)

Nous supposons que les services externes dont le WSIG interagit avec, sont déployés dans des UDDIs publics sur internet. Pour la simplicité de l'étude, nous supposons aussi que nous sommes en train d'interagir avec un seul UDDI. Ainsi, nous constatons que le nombre de messages émis dans la phase de communication $NbMComm$, est un échange de messages synchrones entre les agents de communication et le WSIG. Ainsi, le nombre total de messages est :

$$NbMComm = 2 * NbAgents + NbMServices \tag{4.10}$$

Où NbMServices sont les messages `SOAP` envoyés par le WSIG pour interagir avec les services externes. Nous supposons que NbMServices est égale à 1.

#### 4.6.1.2 Coordination

Dans cette phase, le WSIG envoie la liste des fichiers WSDL des services web susceptibles d'être consommés par les agents de coordination.

Ceci s'explique par les messages $NbMCoor$ suivants :
- $NbMCoor$ messages envoyés par le WSIG aux agents de coordination signalant les services trouvés. Ces messages inclus des informations sur la QoS de ces services, leurs localisations ainsi que des informations sémantiques (IOPE).
- $NbMCoor$ émis par les agents de coordination au WSIG à la réception de chaque service : un `CFP`, un `PROPOSE` et un `REJECT-PROPOSAL` ou `ACCEPT-PROPOSAL` pour chaque agent de coordination.

Ainsi, le nombre de messages émis dans cette phase correspond aux messages envoyés entre le WSIG et les agents de coordination, qui est un échange de messages synchrone :

$$NbMCoor = 2 * NbAgents \tag{4.11}$$

#### 4.6.1.3 Production

Dans la phase de production, le service est enregistré dans l'UDDI locale, ce qui constitue une nouvelle entrée dans le système. Ainsi, le nombre de messages émis dans la phase de production est proportionnel au nombre d'agents de production, et le nombre de nouveaux services enregistrés (nous supposons qu'il y aura un service enregistré à chaque fois). Il s'agit des $NbMProd$ émis par l'agent WSIG pour signaler qu'un nouveau service est intégré, et des $NbMProd$ messages émis pour signaler l'accomplissement de l'action.

Ces messages sont répartis de la façon suivante :
- $NbMProd$ messages émis par le WSIG aux agents de production, pour signaler le début de la phase de production.
- $NbMProd$ messages émis par les agents de production pour extraire les informations relatifs aux services web à intégrer.
- $NbMProd$ messages envoyés par le WSIG pour créer une entrée dans l'UDDI du nouveau service. (4.23.
- $NbMProd$ messages envoyés aux agents de production avec l'adresse du nouveau service.

- $NbMProd$ messages émis par les agents de production au WSIG, signalant la fin des actions de la production.

Ainsi, le nombre des messages émis dans la phase de production est :

$$NbMProd = 5 * NbAgents \tag{4.12}$$

```
private void registerService(WSIGService wsigService) throws Exception {

    if (null != wsigService) {
        log.info("Create new wsig service: "+wsigService.toString());

        // Register wsigService into UDDI
        if (uddiManager != null) {
            ServiceKey uddiServiceKey = uddiManager.UDDIRegister(wsigService);
            wsigService.setUddiServiceKey(uddiServiceKey);
        }

        // Store wsigService into WSIGStore
        wsigStore.addService(wsigService.getServiceName(), wsigService);
    }
}
```

FIG. 4.23 – Enregistrement d'un nouveau service

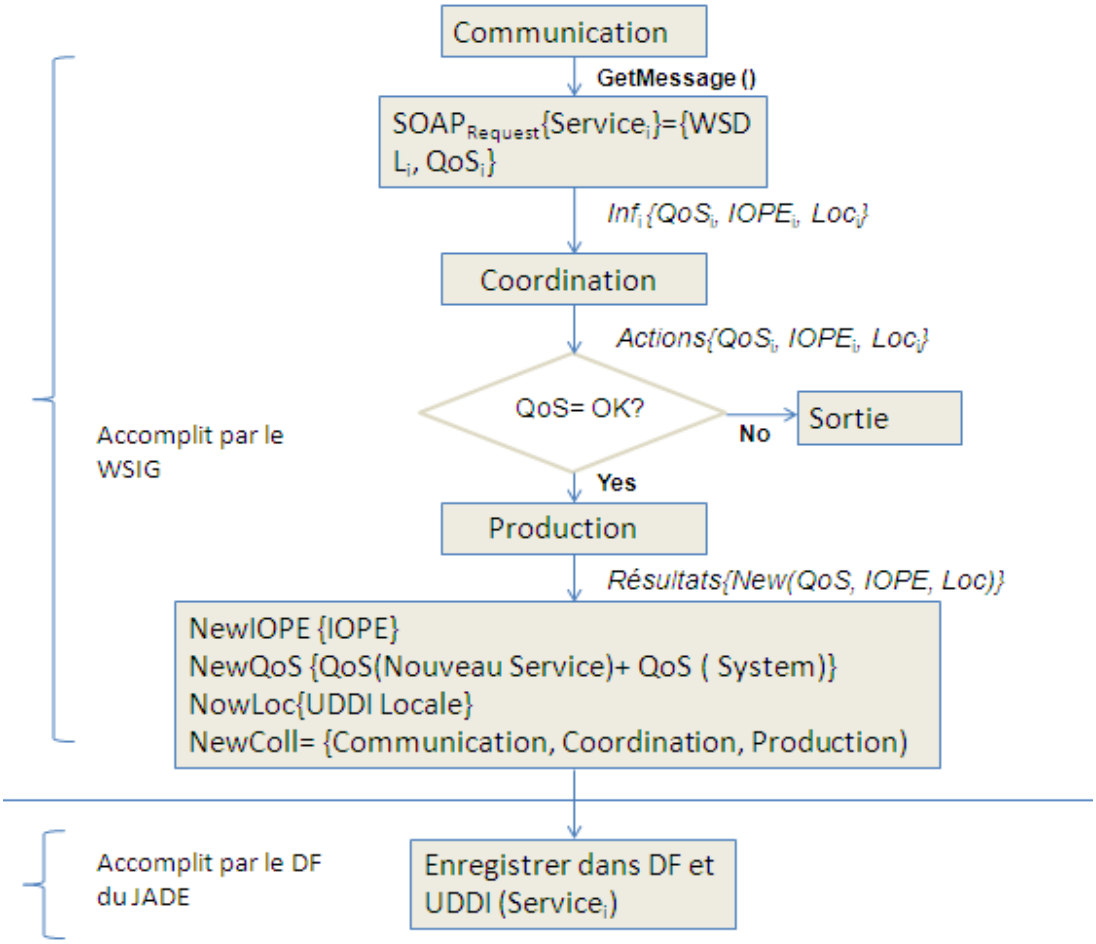



FIG. 4.24 – Orchestration de tâches pour intégrer un service externe

### 4.6.2 Composition de services

Nous décrivons la composition des services telle établie par le processus d'orchestration. Nous déduisons le nombre de messages émis dans chaque phase de la collaboration (Figure 4.26) afin de composer deux services internes.

#### 4.6.2.1 Communication

Dans cette phase, et comme dans le processus d'intégration, les agents de coordination et de production n'émettent pas de messages. Ainsi, la communication est basée sur un échange d'informations non-fonctionnels (QoS) entre deux services internes. Le nombre de messages émis $NbMComm$ dans cette phase est (voir figure 4.26) :

- $NbMComm$ messages `REQUEST` envoyés par les agents de communication au WSIG afin de fournir un service composite entre deux service $i$ et $j$.
- $NbMComm$ messages `INFORM` envoyés par le WSIG aux agents de communication pour confirmer la reception de la demande.
- $NbMComm$ messages `REQUEST` envoyés aux services pour extraire leurs QoS.

Ainsi, le nombre de messages émis dans la phase de communication est :

$$NbMComm = 3 * NbAgents \tag{4.13}$$

#### 4.6.2.2 Coordination

Dans cette phase, l'agent collaboration (WSIG) teste la faisabilité de la composition. Nous supposons que les deux services à intégrer appartiennent à la même classe (communication, coordination ou production). Ceci s'explique par les messages $NbMCoor$ suivants :

- $NbMCoor$ messages émis par le WSIG aux agents de coordination qui interrogent l'ontologie correspondante aux services à composer. Ces messages inclus principalement des informations sémantiques qui définissent les comportements des services (Figure 4.25).
- $NbMCoor$ messages envoyés par les agents au WSIG à la réception des informations sur chaque service : un `CFP`, un `PROPOSE` et un `REJECT-PROPOSAL` ou `ACCEPT-PROPOSAL` pour chaque agent de coordination.

```
public AgentAction getAgentAction(Vector<ParameterInfo> soapParams) throws Exception {

    AgentActionSchema schema = (AgentActionSchema) getOntology().getSchema(getOntologyActionName());
    AbsAgentAction actionAbsObj = (AbsAgentAction) schema.newInstance();
    if (soapParams != null) {
        for (ParameterInfo param : soapParams) {
            String slotName = param.getName();
            AbsObject slotValue = param.getValue();
            AbsHelper.setAttribute(actionAbsObj, slotName, slotValue);
        }
    }
    return actionAbsObj;
  }
}
```

FIG. 4.25 – Interrogation de l'ontologie correspondante des services à composer

Ainsi, nous avons un nombre de message comme suit :

$$NbMCoor = 2 * NbAgents \tag{4.14}$$

#### 4.6.2.3 Production

Dans la phase de production, un nouveau service est composé et enregistré dans le système. Ainsi le nombre de messages émis dans la phase de production est proportionnel au nombre d'agents de production, et le nombre de nouveaux services enregistrés (Nous supposons qu'il y a un service enregistré à chaque fois). Il s'agit des $NbMProd$ messages émis par l'agent WSIG pour signaler qu'un nouveau service est composé et des $NbMProd$ messages émis après accomplissement des actions.

Ces messages sont répartis de la façon suivante :

- $NbMProd$ messages envoyés par le WSIG signalant aux agents de production du nouveau service composite.
- $NbMProd$ messages émis par les agents production pour extraire les informations relatifs au nouveau service.
- $NbMProd$ messages envoyés par le WSIG pour affecter le service à un ou plusieurs agents de production.
- $NbMProd$ messages de fin des actions de la production.

Ainsi, le nombre des messages émis dans la phase de production sera :

$$NbMProd = 4 * NbAgents \tag{4.15}$$

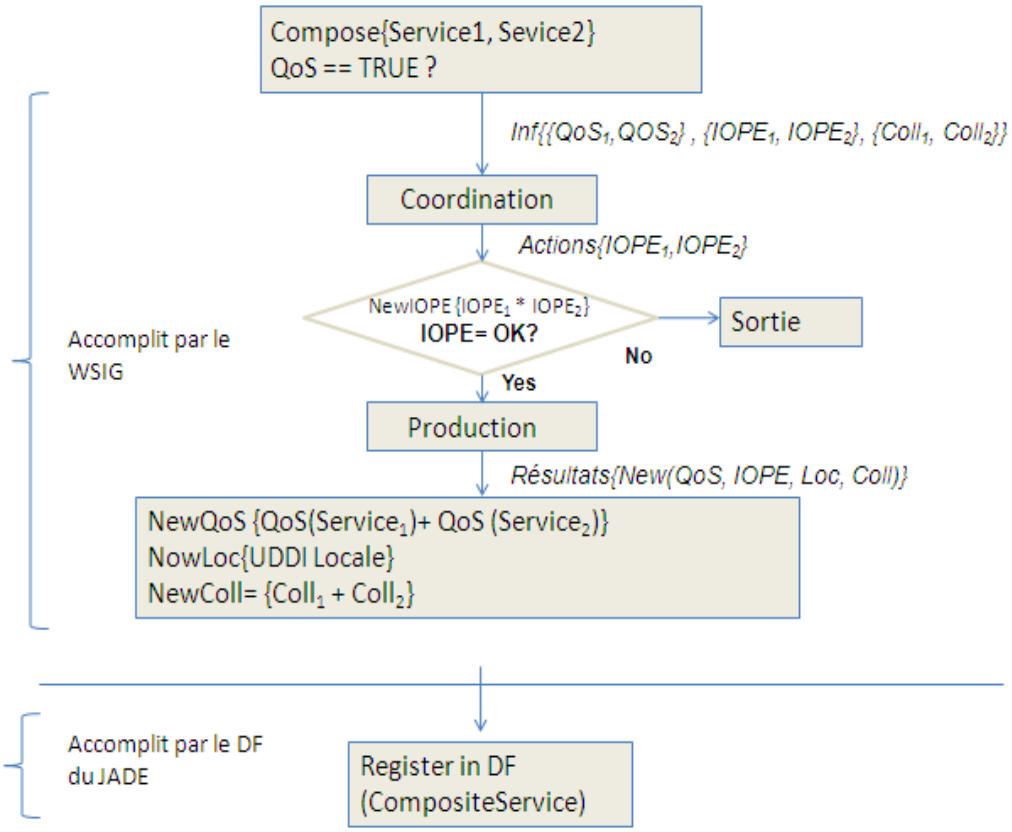



FIG. 4.26 – Processus d'orchestration pour composer deux services internes

Nous résumons dans le tableau ci-dessous, le nombre de messages émis dans chaque phase de collaboration dans le système $U3D$, ainsi que le nombre de messages dans le formalisme SMA-C (qui est le formalisme du modèle C4).

| Espace/Système | SMA-C | $UD3_I$ | $U3D_C$ |
|---|---|---|---|
| Communication | $6 * NbAgents - 2$ | $2 * NbAgents + NbMService$ | $3 * NbAgents$ |
| Coordination | $24 * NbAgents - 16$ | $2 * NbAgents$ | $2 * NbAgents$ |
| Production | $28 * NbAgents - 9$ | $5 * NbAgents$ | $4 * NbAgents$ |

TAB. 4.2 – Comparaison des résultats de SMA-C et $U3D$ dans les trois phases de collaboration

Les données attendues sont les messages émis par chaque agent durant chaque étape. L'échantillon de test est composé d'un nombre d'agents variant de 2 à 35. Un extrait des données récupérées est donné dans les tableaux A.4, A.5 et A.6 qui se trouvent dans les annexes A.

### 4.6.3 Comparaisons de performance

L'étude présenté dans cette section vise à mettre une politique de comparaison entre l'architecture logicielle (SMA-C) (Khezami, 2005) et l'architecture $U3D$ (Cheaib et al., 2008a). Notre objectif dans cette étude est composé en trois parties :

- Reprendre l'étude sur la performance du SMA-C, afin d'avoir de nouveaux résultats intégrant la malléabilité.
- Mettre en œuvre une étude afin d'analyser les interactions dans notre architecture.
- Démontrer qu'avec la nouvelle architecture, le système collaboratif est amélioré en terme d'émissions de messages, en même temps satisfaire la propriété de la malléabilité.

Pour la simplicité, nous nommons l'agent collaboration du SMA-C AC1, et le nouveau agent collaboration AC2. A partir des analyses effectuées, nous pouvons déduire les résultats suivants :

- Dans la phase de communication, le nombre de messages de AC1 équivaut au nombre des agents de communication multiplié par 6. En revanche, le nombre de messages émis pour AC2 dans le processus d'intégration de services est de l'ordre de deux fois plus que le nombre d'agents, ce qui est trois fois moins que les messages émis par AC1. Dans la phase de composition, les messages émis est le nombre d'agents multiplié par trois, ce qui est deux fois moins que l'AC1.

- Dans la phase de coordination, le nombre de messages émis par AC1 correspond au nombre des agents de coordination multiplié par 24. Le nombre de messages émis par AC2 dans le processus de l'intégration est deux fois plus que le nombre d'agents, donc 12 fois moins qu'AC1. C'est le même cas dans le processus de composition, où AC2 envoie 12 fois moins de messages qu'AC1.

- Dans la phase de production, le nombre de messages émis par AC1 correspond aux nombre des agents de production multiplié par 28. Le nombre de messages émis par AC2 dans le processus d'intégration est cinq fois plus que le nombre d'agents, donc 5.6 fois moins que l'AC1. Dans le processus de la composition, le nombre de messages d'AC2 équivaut à quatre fois le nombre des agents, donc sept fois moins qu'AC1.

Dans les figures 4.27, 4.28, et 4.29, nous pouvons voir une comparaison entre la performance du SMA-C, et les résultats attendus de l'agent collaboration (AC2) en termes d'envoie de messages dans chaque phase de la collaboration.

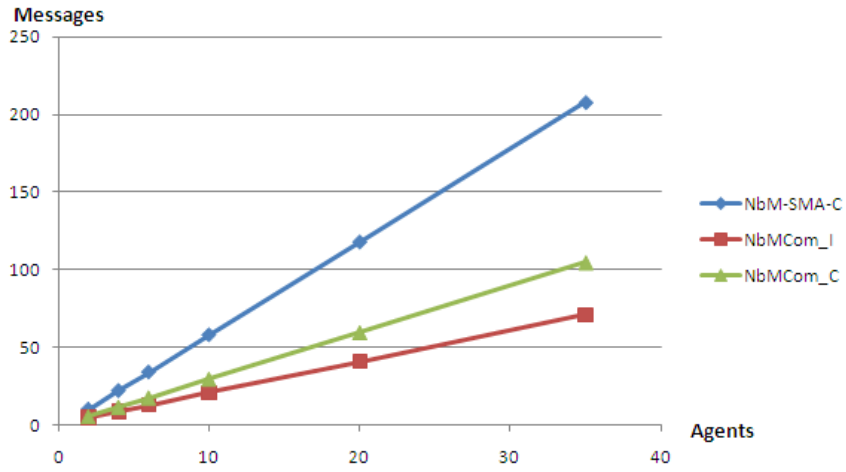



FIG. 4.27 – Comparaison de performance avec l'ancienne et la nouvelle phase (intégration et composition) de communication

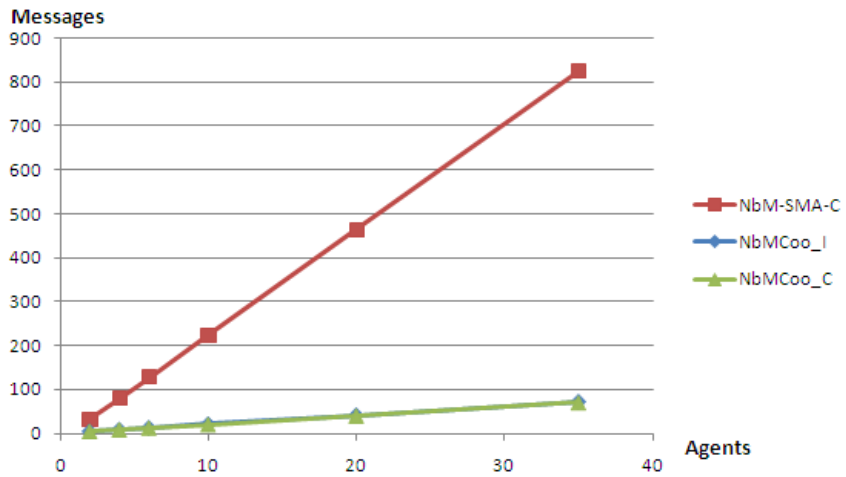



FIG. 4.28 – Comparaison de performance avec l'ancienne et la nouvelle phase (intégration et composition) de coordination

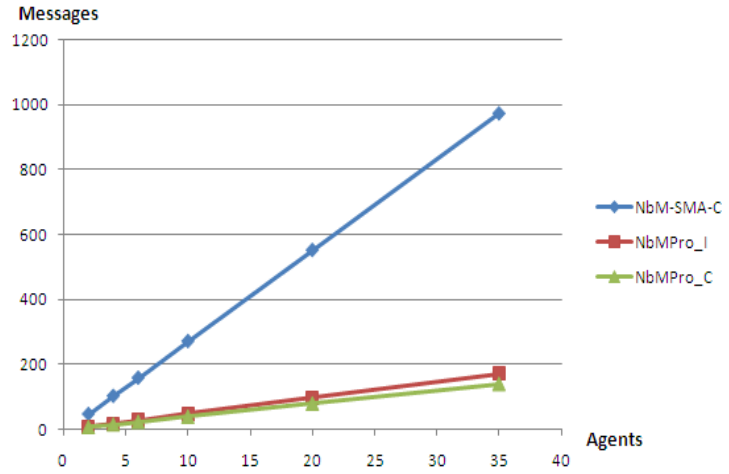



FIG. 4.29 – Comparaison de performance avec l'ancienne et la nouvelle phase (intégration et composition) de production

#### 4.6.3.1 Nombre total de messages

Nous comparons le nombre total de messages (dans toutes les phases de la collaboration) du formalisme SMA-C avec le notre pour les deux processus d'orchestration (l'intégration et la composition). Les résultats sont exploités dans la figure 4.30. Les données de cette étude se trouvent dans les annexes A.

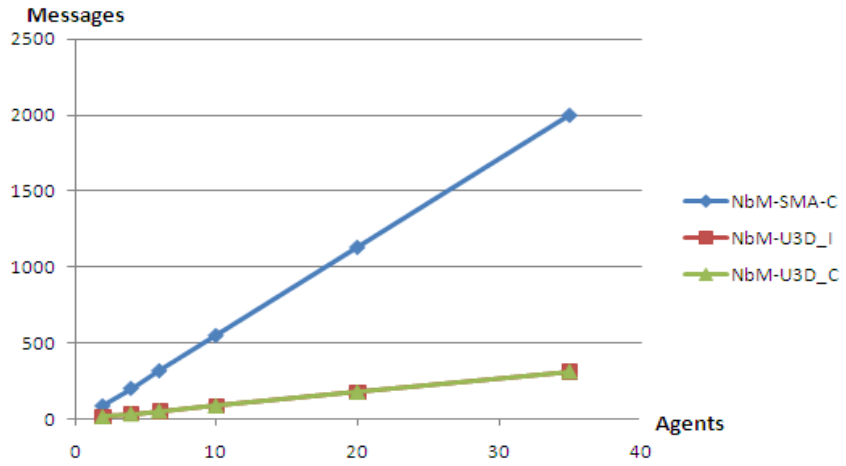



FIG. 4.30 – Nombres de messages totales émis dans les deux systèmes

Nous déduisons le nombre de messages émis par le SMA-C par l'addition de nombres de messages dans toutes les phases de la collaboration. Soit $n$ le nombre d'agents et $F(n)$ la fonction qui calcule le nombre de messages émis. A partir de l'étude de la section 4.5.2.4, le nombre de messages totales de SMA-C est :

$F(n)_{sma(com,coo,pro)} = (6n - 2) + (24n - 16) + (28 * n - 9) = 58n - 27$

Dans l'architecture $U3D$, les trois phases de la collaboration envoient un nombre de messages équivaut à :

- Intégration : $F(n)_{I(com,coo,pro)} = (2n + 1) + 2n + 5n = 9n + 1$

- Composition : $F(n)_{C(com,coo,pro)} = 3n + 2n + 4n = 9n$

En effet, $\forall n > 0 \Rightarrow 9n + 1 > 9n \Rightarrow F(n)_I > F(n)_C$

Supposons que $F(n)_{sma} > 6 * F(n)_I \Rightarrow 58n - 27 > 6 * 9n + 1 \Rightarrow 58n - 54n > 28 \Rightarrow n > 7$

Ainsi, pour $n > 7$, notre système émet 6 fois moins de messages dans les deux phases d'orchestration (intégration et composition).

## 4.7 Conclusion

Dans ce chapitre, nous avons présenté dans la première section les technologies utilisées pour mettre en œuvre notre architecture logicielle. Ensuite, nous avons proposé une étude en UML pour représenter les différentes classes des agents du système. Dans la deuxième section, nous avons présenté une étude pour mesurer la performance des agents dans le formalisme SMA-C (Khezami, 2005). Nous avons appliqué l'architecture $UDDI4C$ comme un support à la malléabilité, afin de calculer le coût de la malléabilité dans le modèle C4 issu du formalisme SMA-C. Ensuite, nous avons présenté une étude pour mesurer la performance de l'architecture $U3D$ en termes de nombre de messages émis.

En ce qui concerne l'étude de l'$UDDI4C$, nous constatons que le coût de la malléabilité en termes de nombre de messages internes échangés est de 15% pour le mécanisme de l'intégration, et de 10 % pour la composition. Evidemment, nous supposons qu'un système fiable est celui qui émet le moins de messages afin d'exécuter un certain nombre d'actions. Ce coût de la malléabilité peut s'avérer important pour les systèmes avec une utilisation limitée ou restreinte, ce qui engendre un goulot au niveau du réseau et du système d'exploitation associé, surtout si la plateforme n'est pas conçu pour exploiter des services sur internet. En même temps, pour les systèmes à larges échelles qui ont une utilisation beaucoup plus importantes, les concepteurs doivent prendre en compte le coût de la malléabilité, où le gain au niveau de la qualité de la collaboration sera plus important que la perte au niveau de la performance du système au niveau du réseau.

En ce qui concerne l'étude de l'$U3D$, les résultats attendus de l'agent collaboration (AC2) nous permettent de mettre en exergue deux points majeurs : La correspondance entre le modèle théorique et les résultats montre que nous pouvons prévoir le nombre de messages échangés, qui seront respectées par l'implémentation, ainsi que la relation entre le nombre de messages et la performance du système qui peuvent être sortie de façon claire. Nous avons remarqué que la performance attendue de la nouvelle architecture est meilleure en termes de nombres de messages émis dans les trois phases de la collaboration, comme nous pouvons le constater dans le graphique représenté dans la figure 4.30. Quoique la production dépende du processus d'orchestration qui met en œuvre une stratégie efficace pour générer de nouveaux services.

En effet, il est assez fréquent que de nombreux services ont des fonctionnalités identiques ou similaires. Il est donc possible que le système génère plusieurs services composites répondant aux préférences des utilisateurs. Dans ce cas, les services composites sont évalués par leurs utilités globales, en utilisant les informations fournis par les attri-

buts non-fonctionnels, dont le poids des attributs est précisé par le demandeur de service. Après qu'un processus composite unique est sélectionné, le service composite est prêt à être exécuté. L'exécution d'un service composite est pensée comme une séquence d'échanges de messages entre les services de bases.

Dans le chapitre suivant, nous proposons deux études de cas : nous montrons comment l'architecture $U3D$ peut apporter de la malléabilité dans le contexte du projet ARITI pour la téléopération collaborative d'un robot, et l'$UDDI4C$ dans le contexte du projet DIGITAL OCEAN pour la collecte des données multimédias.

# Chapitre 5

# APPLICATIONS AUX PROJETS ARITI ET DIGITAL OCEAN

Dans ce chapitre, nous appliquons les deux modèles d'architectures logicielles $U3D$ et $UDDI4C$ sur deux projets différents : Le projet ARITI pour la téléopération de robot, et le projet DIGITAL OCEAN pour la collecte de données multimédias.

Dans le projet ARITI, nous montrons comment notre architecture logicielle $U3D$ peut apporter de la malléabilité en donnant aux utilisateurs la possibilité de créer de nouvelles missions de téléopération dynamiquement. Cette démarche est effectuée en utilisant une ontologie qui décrit les actions que le robot peut effectuer. Ainsi, les agents logiciels vont pouvoir accéder aux descriptions sémantiques de ces actions pour en créer de nouvelles missions de téléopération.

Dans le projet DIGITAL OCEAN, nous montrons comment l'architecture logicielle $UDDI4C$ peut apporter de la malléabilité afin de collecter des données multimédias, ainsi que d'utiliser de divers services web qui agissent comme un support à la collaboration. Nous présentons le collecticiel *OceanydGroupware* qui est développé dans le contexte du projet DIGITAL OCEAN. Ensuite, nous proposons une évaluation subjective de ce collecticiel, afin de mesurer l'impact de la malléabilité mise en place sur les utilisateurs.

Nous procédons par le description du projet ARITI ainsi que le concept de la téléopération.

## 5.1 Le projet ARITI

Le projet ARITI (Augmented Reality Interfaces for Teleoperation via the Internet) a pour objectif scientifique l'étude et la mise en oeuvre de nouvelles méthodes d'assistance au travail et au télétravail collaboratif. Les recherches se situent au niveau de la modélisation, de la conception et de l'évaluation par des expérimentations en situation réelle de ces méthodes. La finalité des recherches concerne, d'une part, le développement d'interfaces

(mono et multi-utilisateurs) à moindre coût moyennant les technologies web et, d'autre part, le développement d'interfaces (mono et multi-utilisateurs) multisensorielles et multimodales utilisant les technologies de réalité virtuelle et augmentée. Les domaines d'applications concernés par le projet ARITI sont : la télérobotique, la télémédecine, l'informatique, la biologie moléculaire, et également tous les secteurs concernés par des activités de groupe permettant de concevoir, de prendre des décisions, d'analyser des problèmes complexes ou de s'organiser.

Dans ce qui suit, nous présentons principalement les interfaces d'assistance à la téléopération et à la téléopération collaborative via le web, autour des plateformes de réalité virtuelle et augmentée.

### 5.1.1 Assistance à la téléopération via le Web

Les premiers travaux du projet ARITI ont permis la mise en place du premier système de téléopération de robot en réalité augmentée via internet en France. Ce système d'assistance au télétravail via le web (figure 5.1) s'est vu attribué le nom du projet (le système ARITI) (Otmane et al., 2000). Il permet d'assister un utilisateur dans la réalisation des tâches de télémanipulation d'objets réels par un robot via un navigateur web. Les techniques utilisées ont permis d'apporter à l'opérateur en situation de télétravail, une assistance à la perception de l'environnement et à la commande d'un robot. Ces assistances ont pour objectif l'amélioration de la précision de la tâche tout en assurant la sécurité durant son déroulement.

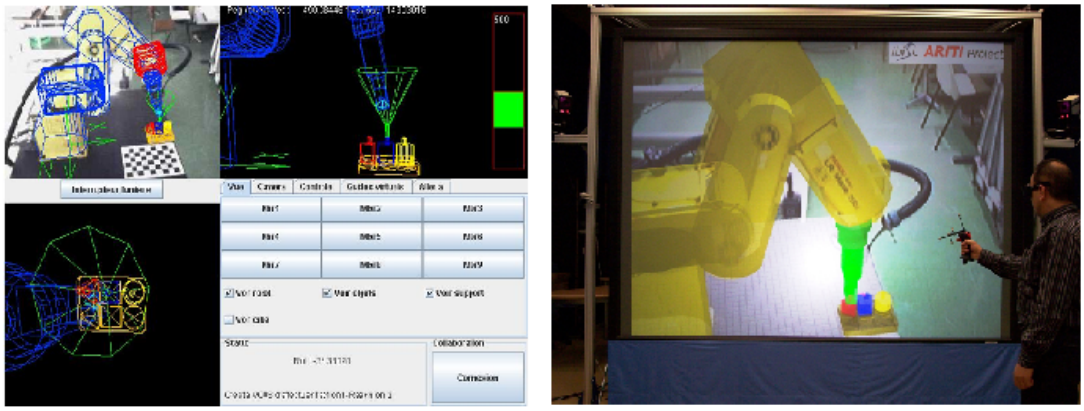


FIG. 5.1 – Capture d'écran d'ARITI-C (à gauche version web et collaborative, et à droite version semi-immervise en réalité mixte)

Les recherches concernant l'assistance au télétravail via le web se sont poursuivies pour proposer des assistances au télétravail collaboratif via le web. En effet, la plupart des missions complexes sont réalisées soit sur site (maintenance industrielle, conception de produits complexes, intervention chirurgicale délicate, etc.), soit depuis un site distant dans le cas du télétravail assisté par ordinateur (téléopération en milieux hostiles, téléchirurgie, télémaintenance, télédiagnostique, etc.). Le succès de certaines missions exige souvent une collaboration avec d'autres personnes (notamment des experts) possédant des compétences et des rôles différents. C'est dans ce contexte que des travaux ont été réalisés pour la conception d'un collecticiel pour l'assistance à la téléopération via le web, ARITI-C (Khezami, 2005). L'objectif de ce travail concerne la modélisation et la

conception d'un système multi-agents de collaboration pour l'assistance à la préparation de missions et à l'exécution des tâches collaboratives. La première application de ce travail est la téléopération collaborative d'un robot via internet en utilisant le système ARITI, couplé avec le formalisme SMA-C. L'interface du collecticiel (figure 5.2) permet à un groupe d'utilisateurs distants de communiquer, de se coordonner et de coopérer avant et pendant l'exécution des tâches.

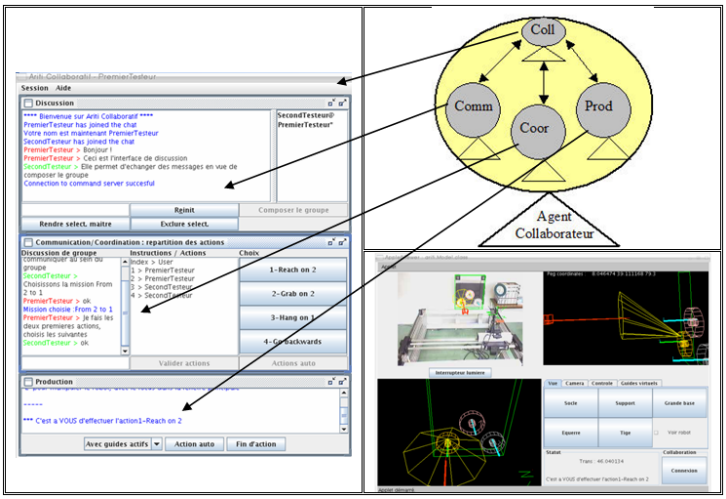



FIG. 5.2 – Capture d'écran des nouvelles interfaces résultantes pour la téléopération collaborative via le web. A gauche, l'interface de l'assistant pour la communication, la coordination et la production. A droite, en haut, l'agent collaborateur (décrit dans le chapitre 1), en bas l'interface du système de téléopération collaborative ARITI-C.

### 5.1.2 Contraintes et solutions

Le projet ARITI présente plusieurs avantages (sans citer sa réalisation technique), l'un d'eux c'est d'avoir une assistance des opérations humaines pour la perception de la scène, la supervision des tâches et le contrôle du robot, principalement par l'insertion des guides virtuels actifs. En outre, ce système présente quelques désavantages, qui se concentrent principalement autour des missions ou des comportements que les utilisateurs peuvent manipuler lors de la téléopération. En effet, les comportements qui permettent la manipulation du robot sont statiques et codées au préalable dans le noyau de l'application. Ainsi, les utilisateurs sont toujours limités à quelques petites manipulations ou comportements (saisie d'objets, rotation etc.). L'insertion de nouveaux comportements lors de la phase de collaboration est quasiment impossible.

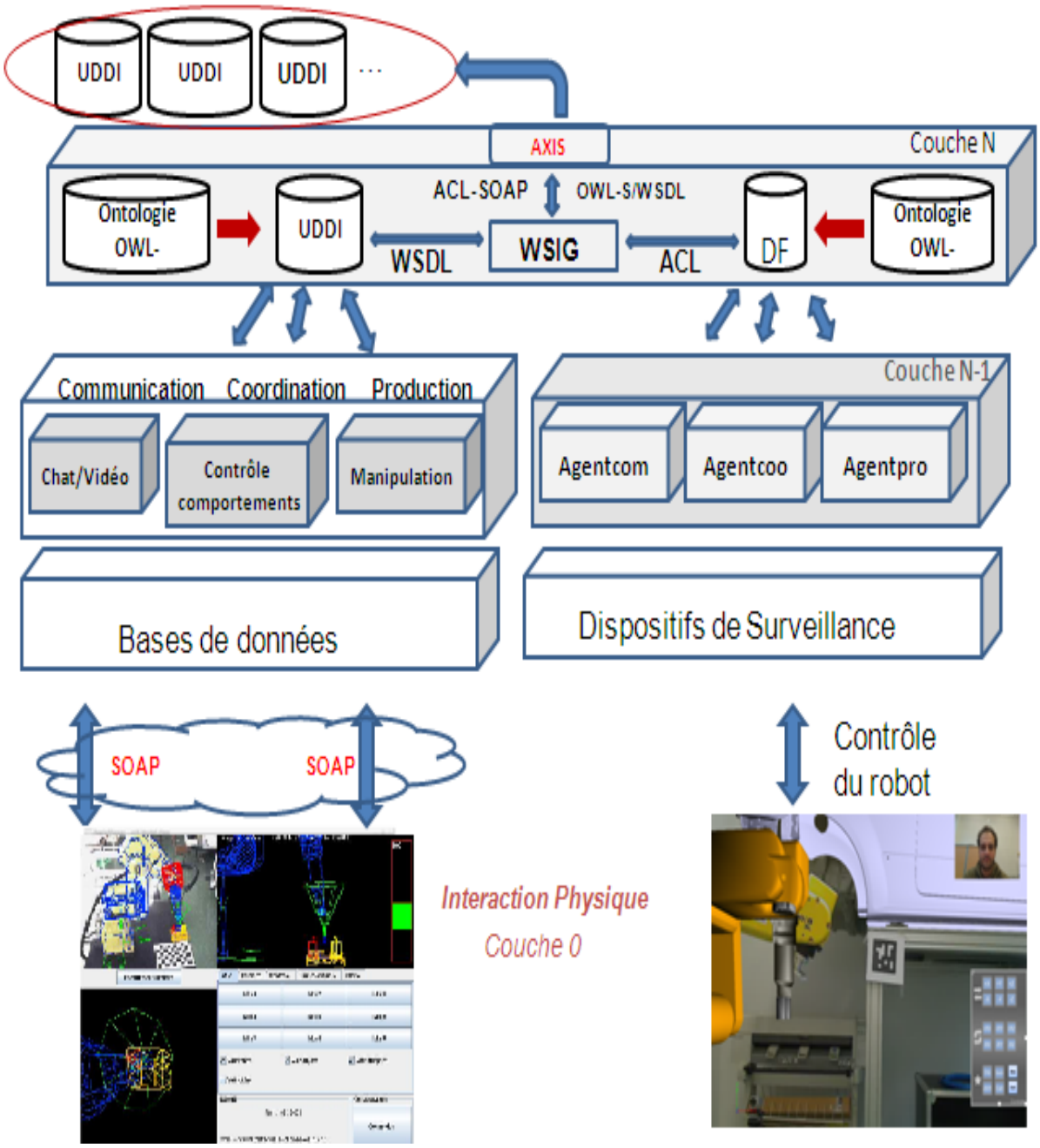



FIG. 5.3 – Application de l'architecture UD3 pour la téléopération du robot

Notre objectif est d'appliquer notre architecture logicielle $U3D$ sur le système ARITI, comme nous pouvons voir dans le figure 5.3. La couche physique est ainsi constituée d'une applet Java pour visualiser le robot virtuel à téléopérer. La communication avec le serveur est basée sur des requêtes SOAP. La couche N-1 contient de divers services pour la communication, la coordination et la production. Pour la communication, nous considérons un mécanisme de conversations textuelles et visuelles (flux vidéo) entre les utilisateurs. Les services de coordination contiennent toutes fonctionnalités pour le contrôle du robot virtuel. Les services de production sont les nouvelles missions crées pour la manipulation du robot. Un environnement JADE gère chaque classe de services avec des classes d'agents de communication, de coordination et de production, respectivement. La couche la plus haute du noyau fonctionnel (couche N) est basée sur des composants pour la traduction

des messages (expliqué dans le chapitre 4). Ainsi, elle sert à lier chaque classe d'agents avec sa classe de services respective. Dans notre travail, nous nous ne sommes pas concentrés sur les couches qui existent entre le noyau fonctionnel et la couche physique. Ces couches peuvent contenir une base de données, des dispositifs de surveillance ou autres ressources pour la téléopération.

### 5.1.3 Composition de missions

Le version collaborative actuelle de ARITI, sait gérer une poignée d'actions servant de briques de base à la composition des missions. Ces missions sont les différentes étapes permettant d'effectuer des trajectoires divers. Elles peuvent être réelles (le robot réel est asservi au robot virtuel) ou simplement virtuelles (manipulation du robot virtuel uniquement). Dans notre travail, nous considérons seulement la manipulation virtuelle du robot.

#### 5.1.3.1 Ontologie et génération de code

Les missions relatives au robot concernant les trajectoires à effectuer sont codés dans le noyau du ARITI. Ainsi, notre but est de générer de nouvelles missions au cours de l'exécution de l'application. Nous nous basons sur des descriptions sémantiques des missions encodés dans une ontologie. Le but c'est de pouvoir accéder à l'ontologie, composer la mission à partir des descriptions qui correspondent à la requête de l'utilisateur, et ainsi générer du code de la mission, qui sera injecté dans les comportements des agents. Nous utilisons la plateforme Protégé [1] pour créer notre ontologie pour les comportements du robot. Un plugin, BeanOntologyGenerator [2] est utilisé afin de générer le code pour la nouvelle mission, et qui crée le lien avec la platforme JADE. Dans la figure 5.4 nous pouvons voir l'ontologie du robot, qui encode les manipulations possibles du robot, qui possède une pince et 4 axes de rotation : les axes 1 et 2 peuvent effectuer des rotations verticales, les axes 3 et 4 effectuent des rotations horizontales.
A partir de l'ontologie du robot, les diverses manipulations sont crées par les utilisateurs en collaboration, pour former une mission complétée par tous les utilisateurs en collaboration. Dans la figure 5.5, nous pouvons voir le code généré pour la nouvelle mission.

[1] http ://protege.stanford.edu/
[2] http ://protege.cim3.net/cgi-bin/wiki.pl ?OntologyBeanGenerator

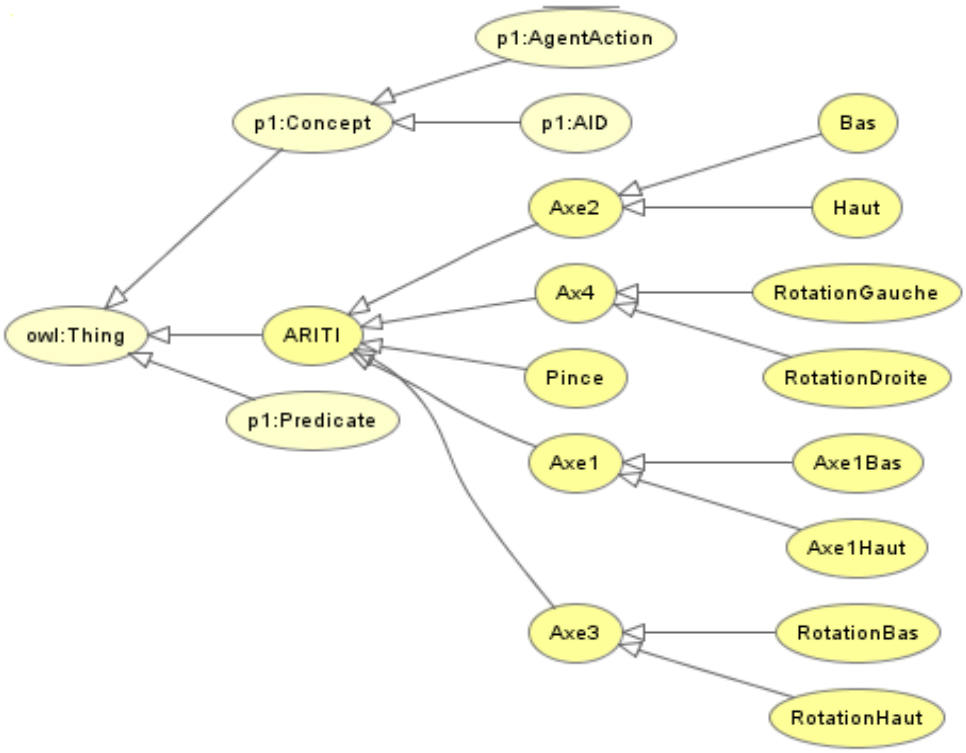



FIG. 5.4 – Ontologie pour les missions du déplacement du robot

```
/** file: TeleoperationOntology.java
 * @author ontology bean generator
 * @version 2010/04/4, 13:40:32
 */
public class TeleoperationOntology extends jade.content.onto.Ontology implements ProtegeTools.ProtegeOntology {
   /**
    * These hashmap store a mapping from jade names to either protege names of SlotHolder
    * containing the protege names.  And vice versa
    */
   private HashMap jadeToProtege;
  //NAME
  public static final String ONTOLOGY_NAME = "teleoperation";
  // The singleton instance of this ontology
  private static ProtegeIntrospector introspect = new ProtegeIntrospector();
  private static Ontology theInstance = new TeleoperationOntology();
  public static Ontology getInstance() {
     return theInstance;
  }
   // ProtegeOntology methods
   public SlotHolder getSlotNameFromJADEName(SlotHolder jadeSlot) {
     return (SlotHolder) jadeToProtege.get(jadeSlot);
   }
   // storing the information
   private void storeSlotName(String jadeName, String javaClassName, String slotName){
       jadeToProtege.put(new SlotHolder(javaClassName, jadeName), new SlotHolder(javaClassName, slotName));
   }
   // VOCABULARY
   /**   * Constructor  */
    private TeleoperationOntology(){
    super(ONTOLOGY_NAME, BasicOntology.getInstance());
    introspect.setOntology(this);
    jadeToProtege = new HashMap();
    try {
float Axes[];
int mission_id;
int status;

Axes = new Get_Axes();
mission_id = new mission();
do
{
    mission_id.move(mission_id, Axes);
    status = mission_id.status();
    }
while (status == 1);
   }catch (java.lang.Exception e) {e.printStackTrace();}
  }
  }
```

*Code généré automatiquement*

*Code ajouté manuellement pour extraire des informations relatives aux missions dans la base de données*

FIG. 5.5 – Génération du code pour une nouvelle mission

#### 5.1.3.2 Interface de collaboration

L'interface présentée dans la figure 5.6 est l'héritage de l'ancien système ARITI-C (ARITI collaboratif) : c'est celle qui s'affiche au lancement de l'applet, elle est le noeud autour duquel sont conçues les autres interfaces pour les fonctionnalités annexes. L'interface est décomposée en trois vues et un panel de contrôle, qui est constitué d'onglets correspondants à diverses fonctionnalités, avec un bouton de collaboration "Normal mode" destiné à ouvrir l'interface de travail collaboratif ancienne. Ainsi, la nouvelle interface contient un nouveau bouton "tailoring mode" qui permet la composition de missions en cours de l'exécution.

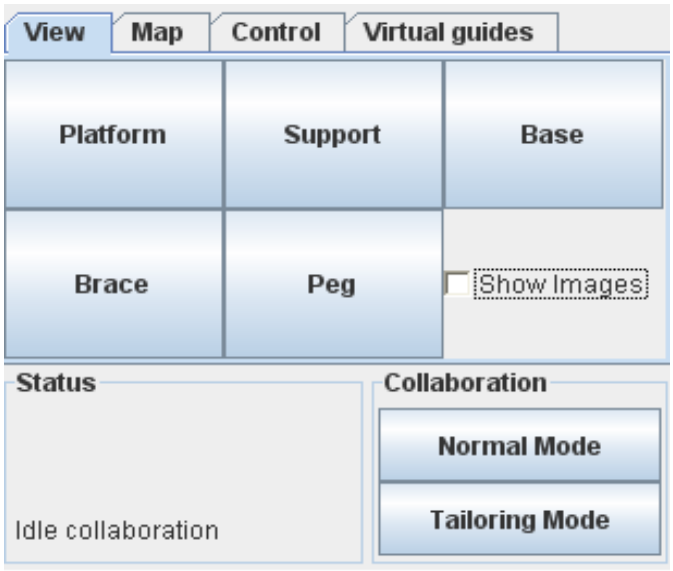



Fig. 5.6 – Nouvelle interface de connexion

Une fois connectée, une interface du robot virtuel est affichée, comme nous pouvons voir dans la figure 5.7. Cette interface permet de créer de nouvelles missions une fois le groupe d'utilisateurs est crée. Ainsi, l'utilisateur "maître" pourra créer ses sous-missions et les diffuser au reste du groupe en utilisant un simple "chat".

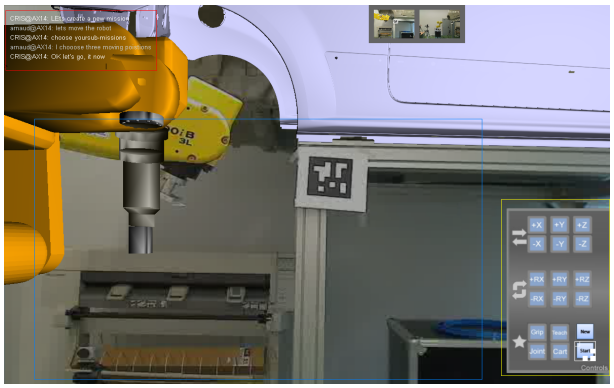


Fig. 5.7 – L'interface graphique pour la téléopération

#### 5.1.3.3 Communication

Une fois le groupe constitué, chaque membre possède une nouvelle interface qui apparaît, constituée d'une zone de discussion intra-groupe (en haut à gauche) et d'une interface de manipulation de robot afin de créer les missions (en bas à droite). Le maître a des boutons de validation ou de choix automatique (aléatoire) de mission.

#### 5.1.3.4 Coordination

Dans la phase de coordination, les utilisateurs créent leurs sous-missions. Ainsi, une mission est l'ensemble des sous-missions (ou actions) définis : à chaque action est associé un bouton. Chaque utilisateur cliquant sur un bouton réserve l'action correspondante si elle n'est pas réservée. Un deuxième clic libère l'action. Chacun est averti des choix des autres utilisateurs par un texte dans l'interface. Le maître ne peut valider que quand toutes les actions sont distribuées. Il peut également choisir d'un clic sur un bouton une répartition automatique des actions. Dans la table 5.8, les sous-missions sont sauvegardées en tant qu'entrées numériques pour la manipulation du robot. Ainsi, une requête est envoyée à l'ontologie qui décrit les actions que le robot est apte à accomplir. Les sous-missions sont ainsi composées par rapport à leurs descriptions sémantiques dans l'ontologie. Une fois la mission est crée, le code est généré pour la nouvelle mission. La validation amène au passage en production.

#### 5.1.3.5 Production

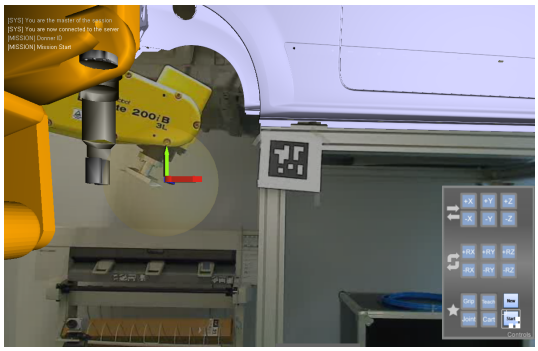
Une fois la mission est confirmée, les sous-missions (Figure 5.8) des divers utilisateurs en collaboration sont composées afin de créer la mission complète (Figure 5.9). Pour l'utilisateur, ce processus est invisible, et la production, pour lui, se déroule au niveau de l'interface principale (Figure 5.10). En utilisant le panel de contrôle du robot ou les raccourcis clavier, l'utilisateur qui fait l'action dirige le robot virtuel. Enfin, une fois toutes les actions terminées, un message invite le maître à réinitialiser la session.

| | 0 : P Initiale | 1 : P Finale |
|---|---|---|
| 17 | -28.927,-5.52635,93.9198 | -18.4624,-5.53018,93.0043 |
| 18 | -20.4564,-6.5287,93.1784 | -11.4837,-13.0243,92.3926 |
| 19 | -20.4095,0.471178,93.6768 | -10.9414,-17.0263,92.8481 |
| 20 | -7.45057,-26.0323,92.5433 | -16.4236,-22.5241,93.3277 |
| 21 | -10.5041,-22.5285,92.1073 | -10.5033,-12.5257,92.1072 |
| 22 | -10.5033,-12.5257,92.1072 | -14.99,-17.0236,92.4997 |
| 23 | -9.50462,-11.0269,92.02 | -21.9663,0.971717,93.1107 |
| 24 | -20.0204,-16.028,92.94 | -12.9971,-16.0245,92.3251 |
| 25 | -31.4354,-13.0203,93.9379 | -17.4862,-13.0306,92.7179 |
| 26 | -17.4835,-4.0308,92.7177 | -29.4434,-4.02608,93.7641 |
| 27 | -33.4301,-23.5258,94.1125 | -20.9751,-23.5262,93.0228 |
| 28 | 5000,5000,5000 | 4993.92,4989.52,5000.61 |
| 29 | -19.5247,-6.52919,92.3947 | -30.9863,-6.52732,93.3975 |
| 30 | -22.5356,-10.0269,92.6582 | -15.5361,-10.0208,92.0453 |
| 31 | -16.0142,-16.5214,92.0872 | -28.9919,-16.5204,93.2226 |
| 32 | -30.9848,-15.0239,93.3971 | -30.9883,-11.0284,93.3978 |
| 33 | -30.9883,-11.0284,93.3978 | -30.9861,-5.52702,93.3976 |
| 34 | -30.9861,-5.52702,93.3976 | -30.9795,5.47554,93.3971 |

FIG. 5.8 – Sauvegarde des coordonnées des déplacements composant une mission

| | 0 : ID Mission | 1 : Type de l'élément | 2 : ID | 3 : Etat |
|---|---|---|---|---|
| 6 | 3 | ARITI_Missions Déplacement | 25 | 1 |
| 7 | 3 | ARITI_Missions Déplacement | 26 | 1 |
| 8 | 3 | ARITI_Missions Déplacement | 27 | 1 |
| 9 | 3 | ARITI_Missions Déplacement | 15 | 1 |
| 10 | 4 | ARITI_Missions Déplacement | 28 | 0 |
| 11 | 5 | ARITI_Missions Déplacement | 29 | 1 |
| 12 | 5 | ARITI_Missions Déplacement | 30 | 1 |
| 13 | 5 | ARITI_Missions Déplacement | 31 | 1 |
| 14 | 6 | ARITI_Missions Déplacement | 32 | 1 |
| 15 | 6 | ARITI_Missions Déplacement | 33 | 1 |
| 16 | 6 | ARITI_Missions Déplacement | 34 | 1 |
| 17 | 7 | ARITI_Missions Déplacement | 38 | 1 |
| 18 | 7 | ARITI_Missions Manipulation | 36 | 1 |
| 19 | 7 | ARITI_Missions Sélection | 35 | 1 |
| 20 | 8 | ARITI_Missions Stops | 38 | 1 |
| 21 | 8 | ARITI_Missions Déplacement | 39 | 1 |
| 22 | 9 | ARITI_Missions Déplacement | 40 | 1 |
| 23 | 9 | ARITI_Missions Déplacement | 41 | 1 |

FIG. 5.9 – Composition de missions à partir des sous-missions



(a) Déplacement utilisateur 1

(b) Déplacement utilisateur 2

(c) Déplacement utilisateur 3

(d) Déplacement utilisateur 4

FIG. 5.10 – Déroulement de la mission composée à partir des 4 sous-missions

Notre système permet également l'intégration de services web externes qui peuvent être exploité dans l'application. A ce jour, les services web publics offrent des fonctionnalités génériques, tels des services de communication (chat) ou de coordination (calendrier), tandis que des fonctionnalités spécifiques pour la téléopération restent très rares. Ainsi, un de nos but est de créer un répertoire UDDI dédié pour la téléopération (UDDIx robotics), et qui offrent des services web tels la vision stéréo, retour haptique, et tout autre services

qui peuvent être exploité dans le cadre de la téléopération. Ces services sont implémentés en Java et enrobé avec une couche XML pour pouvoir interagir avec d'autres machines d'une façon interopérable. Dans notre travail, nous nous sommes concentrés seulement sur la composition de missions dans le système ARITI.

## 5.2 DIGITAL OCEAN : Reconstruction du monde sous-marin

Le projet DIGITAL OCEAN (Dinis et al., 2008) (ANR RIAM 2006-2009) est supporté par l'agence nationale de recherche en France (ANR), et a pour objectif la création d'un nouveau mode de distribution de contenus multimédia. Dans la figure 5.11, nous présentons les différentes aspects du projet qui regroupent ses partenaires. En ce qui nous concerne, nous but est de déployer un collecticiel sur internet, afin de pouvoir collecter des fichiers multimédias (audio, vidéo, images, textes) prises par des plongeurs ou des clubs des plongées. Le but est de charger ses fichiers dans une application 3D hors ligne, qui représente une région du monde sous-marin, afin de pouvoir l'enrichir par des fichiers réelles, et ainsi reconstituer le fond sous-marin en 3D. Nous présentons plus en détails le collecticiel mis en œuvre dans la section suivante.

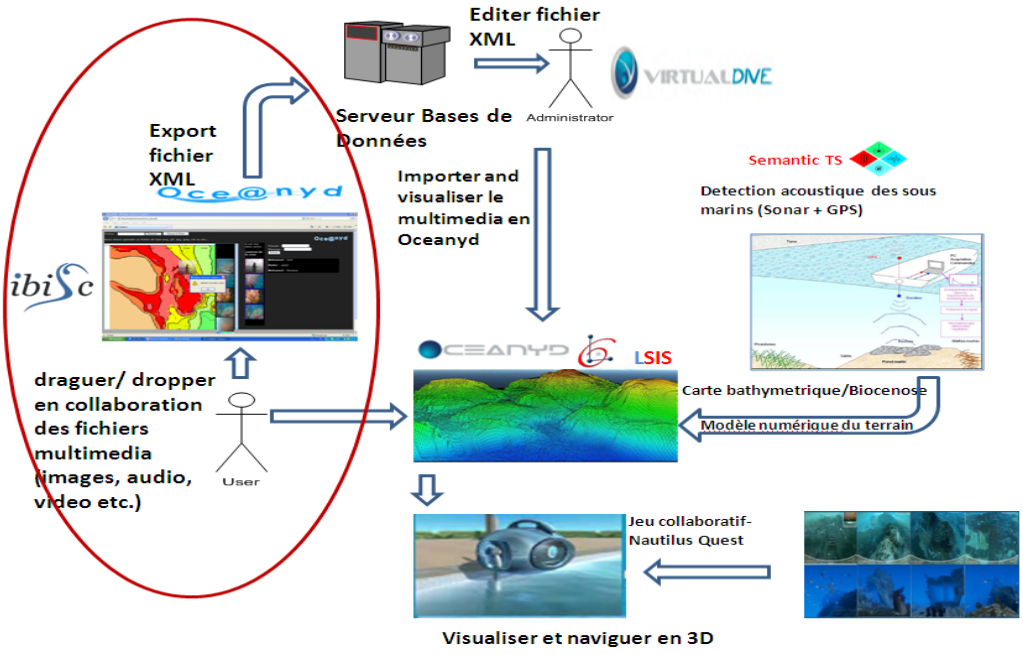



FIG. 5.11 – Scénario d'utilisation - DIGITAL OCEAN

### 5.2.1 Contraintes et solutions

Bien que le projet DIGITAL OCEAN soit destiné à recréer virtuellement l'environnement sous-marin interactif en 3D, il existe plusieurs contraintes avant d'arriver à ce but. Les médias qui sont censés créer ce monde virtuel, sont collectés à travers le collecticiel que nous avons mis en œuvre, *OceanydGroupware* (Cheaib et al., 2009), qui présente une carte bathymétrique en 2D. Ainsi, les utilisateurs utilise une fonction glisser/déposer pour alimenter cette carte par des fichiers multimédias (images, audio, vidéo) pris dans la

région sous-marine représentée par le collecticiel. Ainsi, l'idée est d'importer ces fichiers dans un environnement virtuel, *OceanydExplorer*, que nous avons créé en utilisant le logiciel Virtools, afin d'animer et d'enrichir ce monde sous-marin. Plusieurs contraintes sont à remedier avant d'arriver à notre but :

- Les fichiers collectés par *OceanydGroupware* manquent d'informations nécessaires pour les manipuler dans l'application 3D *OceanydExplorer*, afin de restituer, avec la plus grande précision, leurs localisations géographiques réelles sous-marines. La carte bathymétrique 2D du collecticiel est doté d'une fonctionnalité afin de détecter les coordonnées $(x, y)$ déposées, en utilisant les coordonnées GPS réelles de la région que la carte fournit. Ainsi, ce qui nous manquera comme information est les coordonnée $z$ des fichies déposés.
- Plusieurs informations sont utiles afin d'offrir une documentation efficace des objets déposés par les utilisateurs en collaboration, afin de bien les exploiter dans *OceanydExplorer* qui reconstitue la région de la mer représentée dans la carte dans *OceanydGroupware* avec la plus grande précision.
- Il doit y avoir un lien de communication interopérable entre les deux applications *OceanydGroupware* et *OceanydExplorer*, vu qu'ils sont déployés en utilisant des applications hétérogènes. Le but est de créer un lien bidirectionnelle pour pouvoir, d'une part, charger les fichiers multimédias collectés à travers *OceanydGroupware* dans l'application *OceanydExplorer*, et d'autre part, recharger les informations crées à partir de *OceanydExplorer* dans la base de données qui est commune aux deux applications.

Nous appliquons notre architecture logicielle $UDDI4C$ afin de remédier ces contraintes, comme nous pouvons voir dans la figure 5.12. Ainsi, la couche physique contient les divers interfaces du collecticiel *OceanydGroupware* pour la collecte des données, ainsi que les interfaces de l'application *OceanydExplorer*. Le noyau fonctionnel (sur la couche N-1) contient de divers services classés selon les fonctionnalités qu'ils offrent (expliqué dans la section 5.2.2.1), ainsi que des composants (couche N) pour pouvoir interagir avec des services web externes.

En effet, ça s'est avéré que l'utilisation des services web offre des informations sur la nature des fichiers géolocalisés sur la carte 2D du collecticiel. Un exemple est le site web (www.fishbase.fr) qui contient une base de données sur tous les types de poissons qui existent dans le monde, et des mécanismes afin d'interagir avec ces informations en utilisant les services web. Un autre service web est le CDAS (Coordinated Data Analysis System) par la Nasa, qui fournit des multi-missions simultanées, des instruments de multi-sélection et de comparaison des données venant de leurs diverses missions. Ainsi, les services web proposés offre une interface de programmation distribuée de CDAS. Un troisième site web est le (http ://www.geographynetwork.com), qui est un réseau mondial qui fournit une infrastructure nécessaire pour appuyer le partage de l'information géographique entre les fournisseurs de données, les fournisseurs de services, et les utilisateurs à travers le monde. Grâce à ce réseau, de nombreuses informations géographiques, y compris des cartes dynamiques et des données téléchargeables sont représentées en tant que services web, pouvant être intégrées dans des diverses applications.

Nous interfaçons notre système avec ces sites web (qui agissent comme des UDDIs distribués) afin d'apporter des informations nécessaires pour exploiter les types de fichiers collectés par notre système. Ceci permet une malléabilité au niveau de l'interface et des ressources utilisées par le système, ainsi que de créer un lien entre *OceanydGroupware* et *OceanydExplorer*. Nous présentons plus en details les interfaces de *OceanydGroupware*.

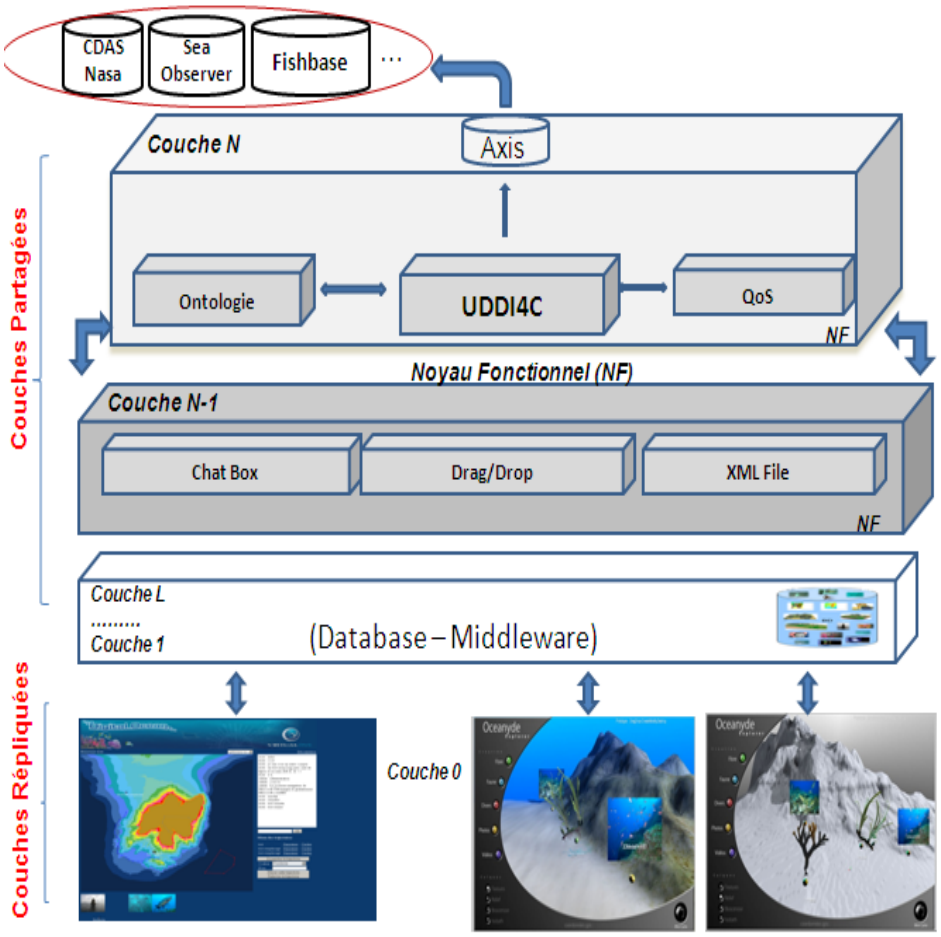



FIG. 5.12 – Application de l'*UDDI4C* sur *OceanydGroupware* et *OceanydExplorer*

### 5.2.2 *OceanydGroupware*

Notre objectif dans ce travail est de traiter des aspects spécifiques du projet telles les collaborations qui peuvent exister entre les utilisateurs, ainsi que les questions concernant l'intégration, l'interopérabilité et bien évidement la malléabilité que l'architecture logicielle peut apporter face aux besoins émergeants des utilisateurs. Comme la nature de la collaboration change continuellement comme une conséquence de l'évolution des besoins de travail, le système doit s'adapter pour prendre en compte l'imprévisibilité des exigences des utilisateurs. Ainsi, apparaît la nécessité d'un nouveau modèle d'architecture logicielle qui satisfait ces exigences dans une application réelle.

Notre système est une application client-serveur déployé en utilisant la plateforme de developpement Netbeans [3]. Les deux couches partagées du noyau fonctionnel sont

[3]http ://www.netbeans.org

déployées sur le serveur, tandis que les autres couches sont répliquées sur chaque machine client. La communication client/serveur est basée sur un flux de réseau. Nous pouvons voir l'interface principale de *OceanydGroupware* dans la figure 5.13 ci-dessous.

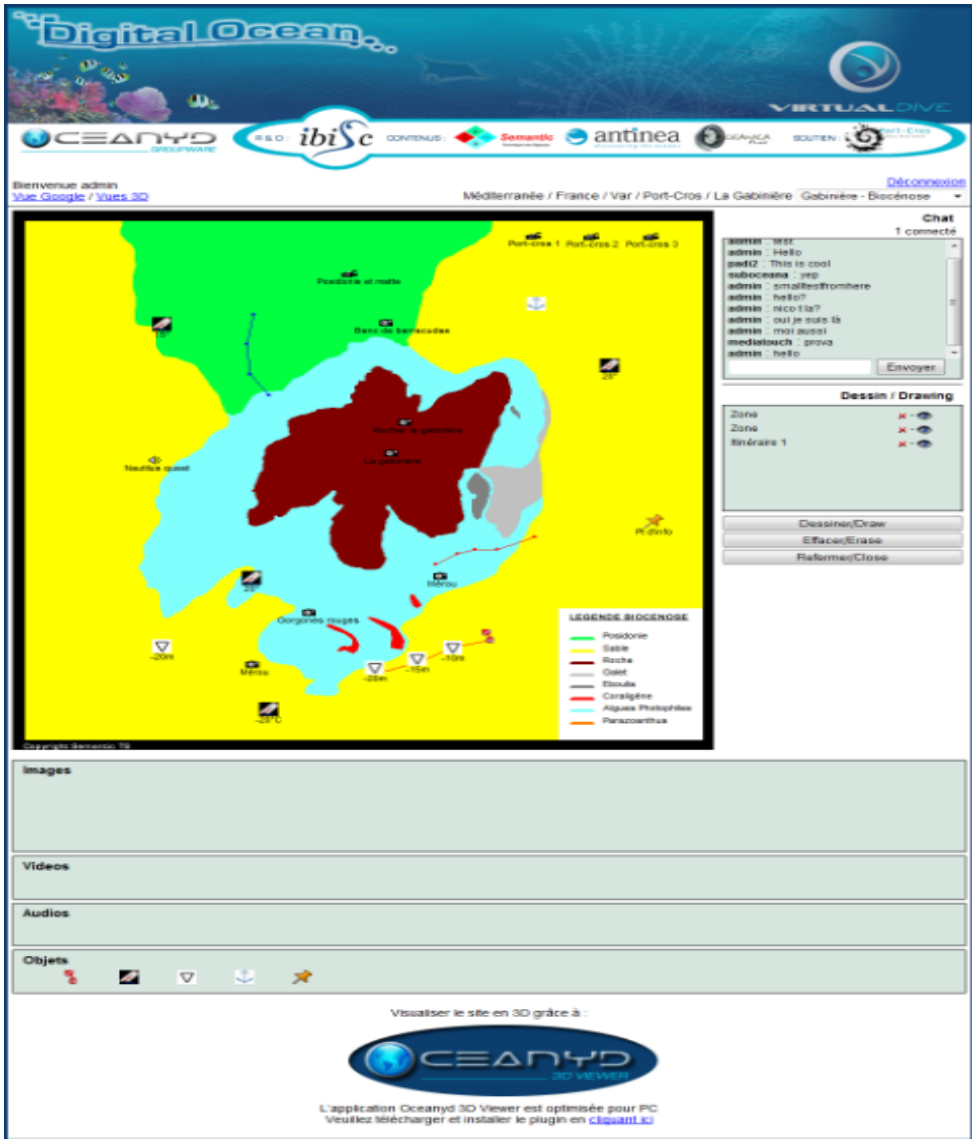



FIG. 5.13 – Page principale de *OceanydGroupware*

#### 5.2.2.1 Couche Physique - IHM

La couche physique contient une interface web manipulée par les utilisateurs en collaboration. Cette interface permet aux utilisateurs de glisser/déposer des fichiers multimédia sur une carte bathymétrique qui représente une région sous-marine (pour notre étude de cas, nous présentons le site de la Gabinière à Port-Cros, au sud de la France) fournie par un partenaire impliqué dans le projet, en utilisant des capteurs, de caméras et d'un GPS. L'objectif est de donner les moyens à des plongeurs professionnels ou des clubs de plongées d'enrichir la carte avec les fichiers multimédias pris dans le même lieu affiché sur la carte.

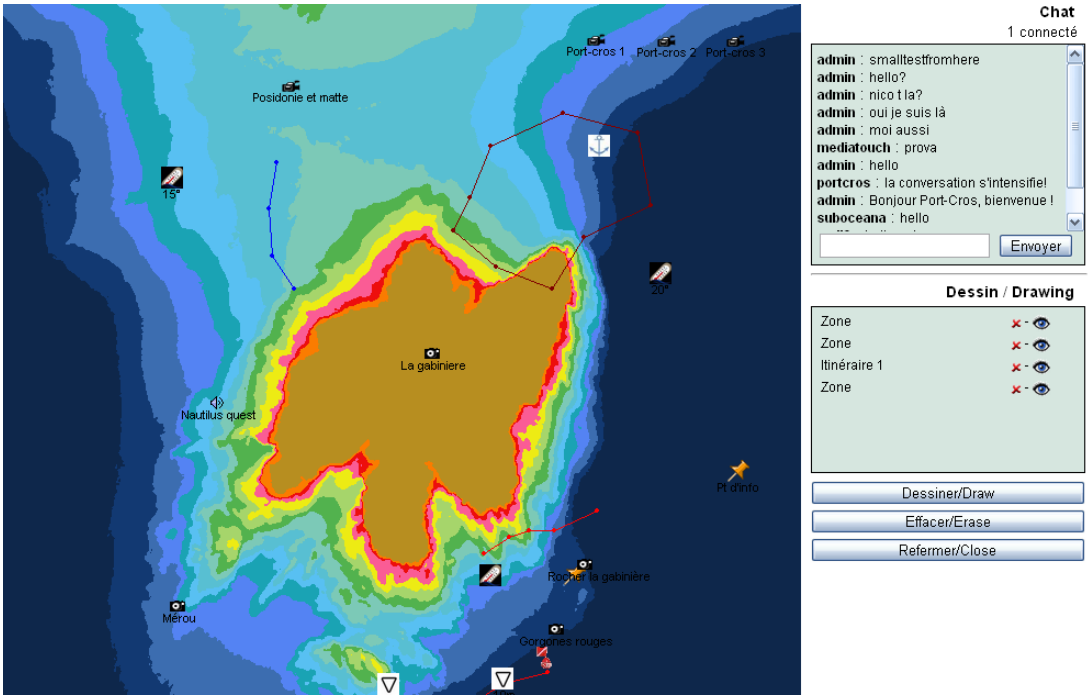



FIG. 5.14 – Communication, coordination et production dans *OceanydGroupware*

La couche physique propose trois fonctionnalités, chacune dédiée à un aspect du modèle 3C, comme nous pouvons voir dans la figure 5.14 :

- Communication : l'application fournit un mécanisme de chat permettant aux utilisateurs d'échanger des informations sur les fichiers échangés, ainsi que leurs expériences d'utilisations de l'application.

- Coordination : Une fonctionnalité est encapsulée dans le système qui divise la carte en un nombre variables de zones, et qui permet de détecter les coordonnées du fichier glissé sur une zone particulière de la carte (le nombre de zones divisant la carte est définit par les utilisateurs en session de collaboration, selon la carte et les tâches à faire, ce qui assure une certaine "adaptabilité du système" au niveau de la coordination des tâches). En plus, l'application fournit des outils pour "dessiner" sur la carte des trajectoires que les utilisateurs ont effectués en plongeant dans cette région de la mer. L'idée est donc de coordonner des tâches qui servent à peupler des régions de la carte, selon les expériences de chaque utilisateur.

- Production : le système implémente un mécanisme d'extraction d'informations sur les fichiers multimédias chargés par les utilisateurs, en créant un fichier XML contenant tous les données nécessaires à chaque fichier glissé sur la carte : date / heure, nature (image, vidéo ou de texte), description, et coordonnée du fichier, où l'utilisateur est invité à entrer les coordonnées exactes de du fichier (x, y, z), à l'endroit où la photo a été prise. Dans le cas où les coordonnées des fichiers sont manquantes, le système affecte automatiquement des coordonnées (x, y), à l'endroit où l'utilisateur a glissé le fichier sur la carte 2D (en utilisant des coordonnées GPS réelle de la région sous-marine). La troisième coordonnées (z) est affectée après avoir chargé les fichiers dans *OceanydExplorer*, à partir de diverses informations récupérées sur internet.

### 5.2.3 Composition de services

Nous mettons en œuvre la composition de services élémentaires offerts par *Oceanyd – Groupware*, afin de générer des services composés qui répondent le mieux aux besoins des utilisateurs. Les services élémentaires proposés par le système sont les suivants :

- Dessiner un segment qui correspond à deux points reliés par une ligne, qui représente une simple trajectoire faite par un seul utilisateur.
- Colorier les trajectoires pour préciser leurs profondeurs dans la mer.
- Lire de vidéos, écouter des fichiers audio et visualiser des images collectées par le collecticiel.
- Afficher des commentaires textuels des utilisateurs sur chaque fichier collecté.
- Communiquer avec d'autres utilisateurs par un mécanisme de Chat.

A partir de ces services élémentaires, les utilisateurs peuvent composer d'autres services afin d'avoir une meilleure précision du processus de collecte de données, ainsi que la capture des expériences des utilisateurs. Pour l'instant, les services composés sont les suivantes :

- Composition de segments : les utilisateurs peuvent composer les segments pour obtenir une trajectoire complète définissant les trajectoires de tous les utilisateurs. Cette trajectoire est sauvegardée dans la base de données en tant que des coordonnées (x,y,z). Ainsi, nous obtenons une meilleure visualisation de trajectoires faites par plusieurs utilisateurs en collaboration.

- Composition de surfaces : une surface est obtenue par la composition de divers trajectoires avec le service de coloriage, qui attribut une couleur qui définit les profondeurs de ces trajectoires respectives. Nous pouvons voir une capture d'écran des deux mécanismes dans les figures 5.15 et 5.16.

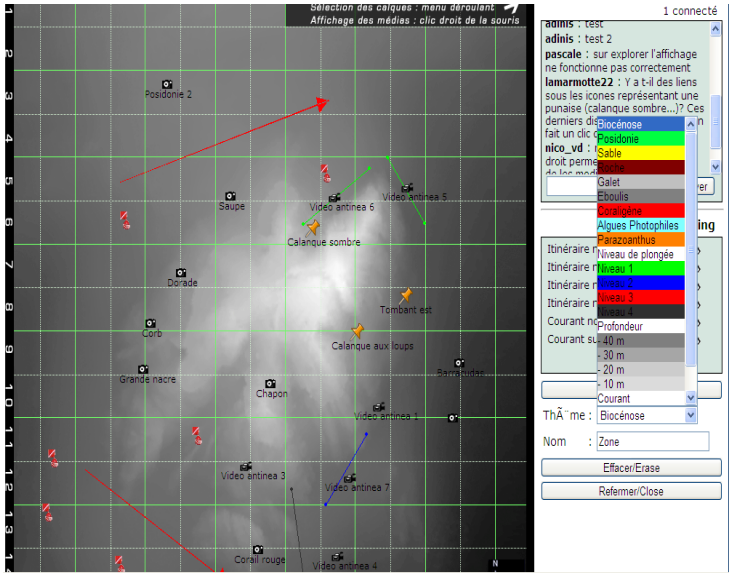



FIG. 5.15 – Services élémentaires dans *OceanydGroupware*

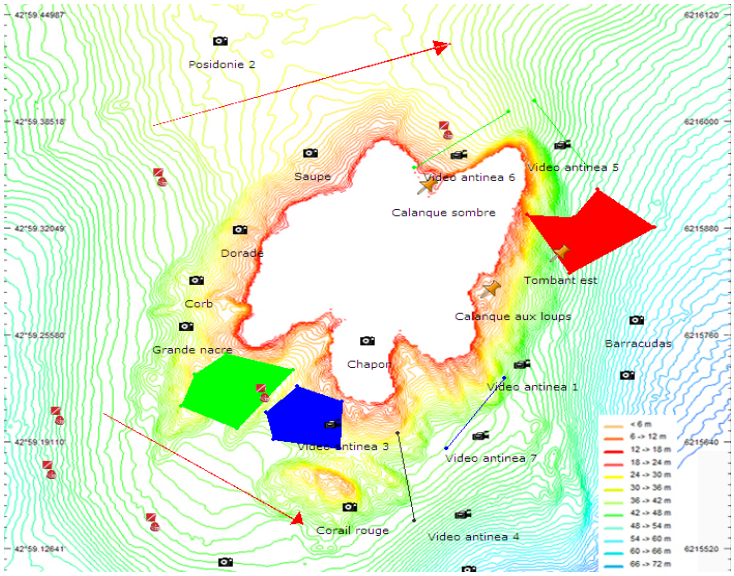



FIG. 5.16 – Composition de surface dans *OceanydGroupware*

### 5.2.4 Intégration de services

Nous pouvons voir un scénario d'utilisation de l'application pour l'intégration de services dans la figure 5.17.

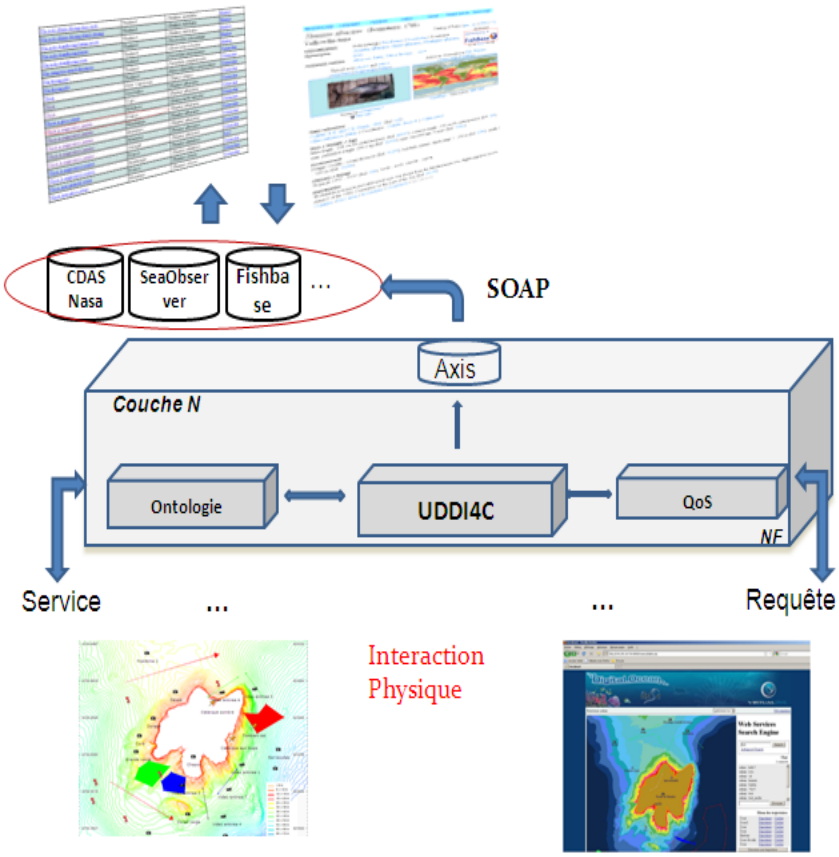



FIG. 5.17 – Connexion aux UDDIs publics pour extraire des informations

Nous discutons principalement le noyau fonctionnel du modèle avec la couche physique d'interaction, ainsi que les services qui sont offerts aux utilisateurs. Les autres couches hébergent la base de données et toutes autres ressources nécessaires pour la cohérence du système. A partir des informations récoltées avec *OceanydGroupware*, un moteur de recherche de services web est intégré dans l'application comme nous pouvons voir dans la figure 5.18. Ce moteur de recherche est interfacé avec des UDDIs connus, qui offrent des informations nécessaires pour documenter les informations collectés. Avec ce mécanisme, l'utilisateur spécifie un mot clé d'un service web qui veut utiliser. L'utilisateur est dirigé vers une liste de services web qui répondent à cette requête. Cette procédure est effectuée en construisant une requête SOAP à partir des descriptions des utilisateurs. Le but de ce processus est divisé en deux points essentiels :

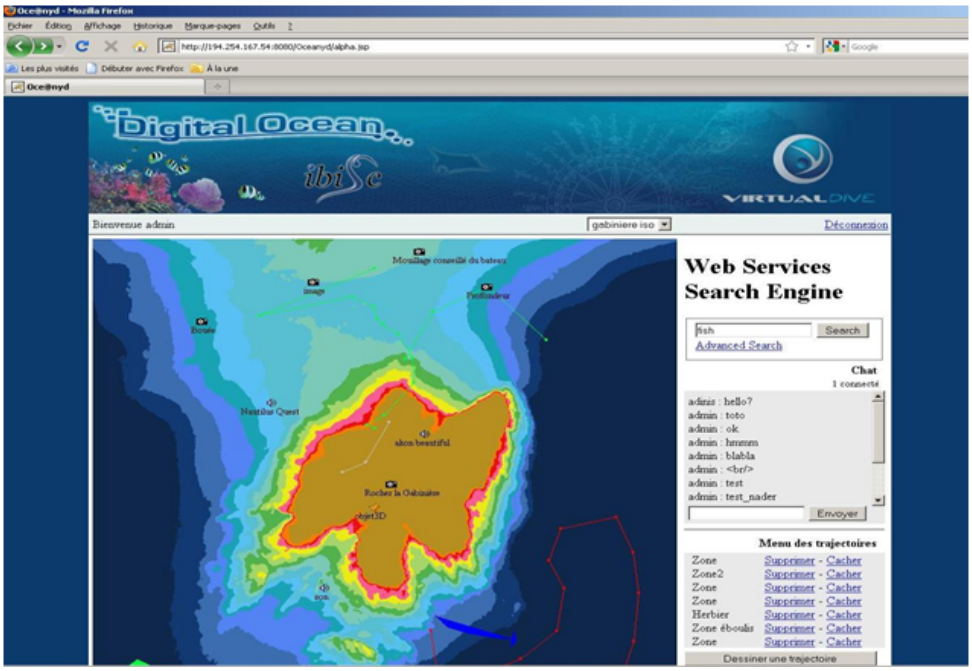



FIG. 5.18 – Moteur de recherche de services web

Le premier est de pouvoir déposer les fichiers collectés dans *OceanydExplorer* (Figure 5.19). En utilisant cette application, les utilisateurs pourront naviguer afin de positionner leurs fichiers dans l'endroit le plus précis possible de l'endroit réel où le fichier à été pris. Nous utilisons le site web (fishebase.org) pour obtenir de diverses informations sur les fichiers collectés et leurs positionnements. Prenons un scénario où l'utilisateur dépose une image d'un poisson "thon". L'utilisateur déclenche le mécanisme de recherche d'informations sur ce type de poisson. A partir de ça, le système se connecte à l'UDDI du site, retrouve le type de poisson (Figure 5.20), ainsi que les informations relatives (Figure 5.21). Ces informations sont ensuite extraite et chargées dans un fichier XML afin de les importer dans *OceanydExplorer*. Ces informations seront associées à l'image du poisson "thon", importée dans l'application (Figure 5.19).

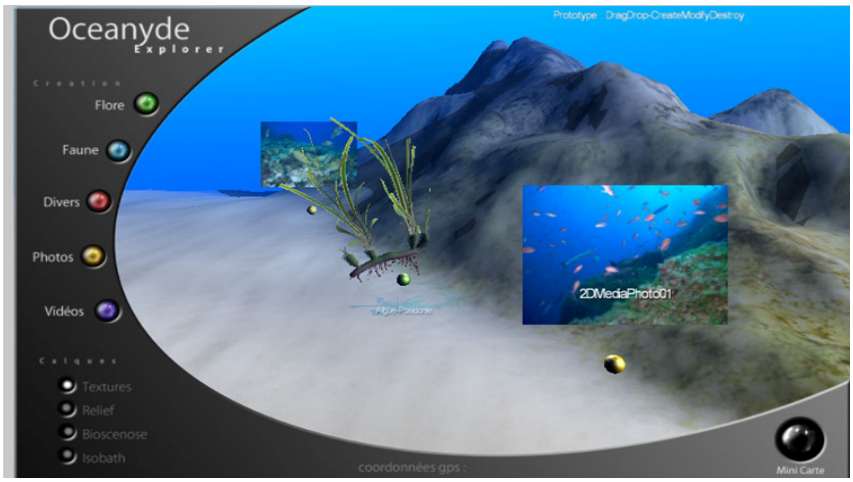



FIG. 5.19 – Repositionnement des médias en utilisant des informations réelles

| | | | |
|---|---|---|---|
| Pla nok-khun-thong-hao-nok | Thailand | *Cheilinus undulatus* | Market |
| Pla nok-khun-thong-klad-deang | Thailand | *Cheilinus trilobatus* | Market |
| Pla nok-kunthong-hang-wow | Thailand | *Cheilinus trilobatus* | Market |
| Pla nok-kunthong-kiew | Thailand | *Choerodon schoenleinii* | Market |
| Pla nok-kunthong-som | Thailand | *Choerodon robustus* | Market |
| Pla sang ka ward thong to | Thailand | *Pseudolais micronemus* | Vernacular |
| Pla thong plu | Thailand | *Parachela oxygastroides* | Vernacular |
| Pla thong plu | Thailand | *Macrochirichthys macrochirus* | Vernacular |
| Thon | New Caledonia | *Chanos chanos* | Vernacular |
| Thon | Seychelles | *Thunnus albacares* | Vernacular |
| Thon | Togo | *Thunnus albacares* | Vernacular |
| Thon à gros yeux | Maldives | *Gymnosarda unicolor* | Vernacular |
| Thon à nageoires jaunes | France | *Thunnus albacares* | Vernacular |
| Thon à nageoires jaunes | Mauritania | *Thunnus albacares* | Vernacular |
| Thon à nageoires jaunes | Mauritius | *Thunnus albacares* | Vernacular |
| Thon à nageoires jaunes | Mauritius | *Thunnus albacares* | Vernacular |
| Thon à nageoires jaunes | Senegal | *Thunnus albacares* | Vernacular |
| Thon à nageoires noires | France | *Thunnus atlanticus* | FAO |
| Thon à nageoires noires | Martinique | *Thunnus atlanticus* | Vernacular |
| Thon aux grands yeux | France | *Thunnus obesus* | Vernacular |
| Thon aux gros yeux | Mauritius | *Thunnus obesus* | Vernacular |

FIG. 5.20 – Liste des poissons thon dans fishbase.org


FIG. 5.21 – Informations sur le poisson déposé par l'utilisateur en tant qu'image

La deuxième raison d'utilisation des services web, est de pouvoir intégrer de nouvelles fonctionnalités au cours de l'exécution sans arrêter la collaboration entre les utilisateurs. Comme nous avons discuté auparavant, il existe une grande infrastructure sur internet qui offre une grande variété de services. Ces types de services sont des services web géographiques qui utilisent des données et des fonctionnalités connexes pour effectuer des tâches élémentaires de traitement des données géographiques : des conversions d'adresses, des recherches spatiales, du routage, de la cartographie, du géocodage, etc. Ces services web sont exploités dans notre système afin de donner aux utilisateurs de nouvelles ressources qui améliorent la collaboration, grâce à l'utilisation "au cours de l'exécution" des services offerts à la demande.

### 5.2.5 Evaluation de l'architecture logicielle

L'évaluation est importante dans le processus du développement du système afin de démontrer s'il fournit des besoins fonctionnels et non-fonctionnels, et de détecter les failles. Il existe certains nombre d'approches pour la validation et le test du système. Beaucoup de ces techniques, comme par exemple l'inspection du programme, se base sur l'analyse du code source. Cependant, quand les services sont offerts d'un fournisseur externe, le code source de l'implémentation du service n'est pas fournit. En conséquence, le test des systèmes basés sur des services ne peut pas utiliser ces techniques. Les testeurs du système peuvent faire face à quelques difficultés lors des tests des services :

– les services externes sont sous le contrôle du fournisseur des services et non l'utilisateur de ces services. Ainsi, le fournisseur peut supprimer ou modifier ces services à n'importe quel moment, ce qui peut nuire à tout test effectué au service initial. Ces problèmes sont gérés, traditionnellement par des composants logiciels qui main-

tiennent différents version du composant. Actuellement, il n'existe pas de standards afin de gérer les versions de services.

– La vision à long-terme d'une architecture orientée service (SOA), est que les services soient attachés dynamiquement à des applications orientées services. Ceci signifie qu'une application peut ne pas toujours utiliser le même service à chaque fois qu'elle est exécutée. Ainsi, des tests peuvent réussir lorsqu'une application est attachée a un service particulier, mais ne pouvant pas garantir que ce service va être utilisé lors de l'exécution du système.

– Le comportement non-fonctionnel du service n'est pas simplement dépendant de la manière dont il est utilisé par l'application qui est en train de se tester. Un service peut avoir une bonne performance lors de la phase de test parce qu'il n'est pas opéré sous une lourde charge. En pratique, le comportement du service observé peut diffèrer selon les demandes des utilisateurs.

– Le mode de paiement des services peut être très coûteux pour le test des services. Il y a des services qui sont disponibles gratuitement, d'autres payant par forfait, et d'autres à chaque fois qu'il est invoqué. Si les services sont gratuits, alors il se peut que le le fournisseur de ces services ne veut que ces services soient chargés par des applications en phase de test.

– La notion de compensation des actions doit être prise en compte. Celle ci est invoquée lorsqu'une exception survient et des engagements antérieurs qui ont été faites doivent être révoqués. Il existe des problèmes en testant de telles actions, car ils dépendent des échecs des autres services. Faire en sorte que ces services échouent durant le processus de test peut être très difficile.

#### 5.2.5.1 Evaluation des systèmes collaboratifs

L'évaluation des systèmes collaboratifs consiste à estimer les performances du système par rapport à un ensemble d'exigences et aux besoins des utilisateurs. Il existe plusieurs méthodes et techniques pour évaluer ces systèmes. Ainsi, nous avons classé ces techniques sommairement de la façon suivante ; Les techniques qui utilisent des :

- Evaluations heuristiques
- Tests utilisateur
- Expériences en laboratoire
- Interview et questionnaire
- Simulation
- Scénarios
- Analyse de protocoles et inspection

Il existe d'autres méthodes d'évaluations. La plus élémentaire consiste à effectuer des calculs statistiques. En effet dans toute évaluation, savoir qui utilise le système, à quelle fréquence et dans quel contexte peut s'avérer très pertinent. Un autre type de méthode consiste à calculer des coefficients d'évaluations à partir de la valeur-ajoutée apportée par

la collaboration du point de vue économique. Les indicateurs les plus souvent utilisés sont la qualité du produit, le temps d'exécution et le coût. Evidemment, il arrive souvent de combiner ces différentes méthodes pour avoir un résultat plus probant.

Un des critères les plus pertinents pour l'évaluation d'un système collaboratif est son utilisabilité. Celle-ci désigne la qualité d'une application qui est facile et agréable à utiliser et à comprendre. L'utilisabilité d'une application se mesure à trois grands critères objectifs :

- L'efficacité, c'est-à-dire l'atteinte des objectifs par l'utilisateur.
- L'efficience, c'est-à-dire l'économie des ressources nécessaires pour atteindre ces objectifs.

Une application est donc utilisable si l'utilisateur peut réaliser sa tâche (efficacité), qu'il consomme un minimum de ressources pour le faire (efficience) et que le système est agréable à utiliser (satisfaction). D'autres aspects peuvent entrer en ligne de compte pour évaluer l'utilisabilité générale de l'application, comme la sécurité ou maîtrise (le nombre d'erreurs commises par l'utilisateur et la rapidité de corrections de ces erreurs) et la facilité d'apprentissage (la compréhension correcte et l'assimilation rapide du mode de fonctionnement). Le concept d'utilisabilité a donné naissance à son propre instrument de mesure : les tests d'utilisabilité. Il ne s'agit pas de la seule méthode pour apprendre ce que les utilisateurs pensent de l'application (citons également l'évaluation experte, les interviews, les focus groupes, les données d'usage quantitatives ou qualitatives), mais c'est probablement la plus appropriée. En effet, tandis que les interviews ou les focus groupes permettent de recueillir l'opinion des utilisateurs, le test d'utilisabilité permet d'observer directement le comportement de l'utilisateur face à l'application et d'identifier, ainsi, concrètement les problèmes qu'il rencontre. Ainsi, le meilleur moyen de savoir si ce que nous proposons à un utilisateur lui convient, c'est de le lui demander, et le meilleur moyen pour lui de répondre à la question, c'est de l'essayer.

L'objectif du test d'utilisabilité est de demander à un petit nombre d'utilisateurs représentatifs de réaliser quelques tâches type de l'application. C'est en observant l'utilisateur en situation que nous pourrons relever les difficultés qu'il rencontre, les erreurs qu'il commet, les questions qu'il se pose et les fonctionnalités qu'il apprécie ou non.

#### 5.2.5.2 Evaluation de *OceanydGroupware*

Le test d'utilisabilité requiert des ressources matérielles limitées, mais une préparation soignée. Outre la désignation de l'observateur et le recrutement des utilisateurs représentatifs, les ressources suivantes sont indispensables pour conduire un test d'utilisabilité :

- Identifier et recruter les utilisateurs représentatifs.
- Identifier les tâches représentatives.
- Un scénario, dont le but est de garantir que tous les participants seront traités sur un pied d'égalité. Le scénario doit comprendre une explication des objectifs du test aux participants, une description du déroulement du test et des tâches à effectuer, les consignes à donner aux participants ainsi que le questionnaire qui leur sera remis après le test.

- Un questionnaire pré-évaluation, pour s'assurer que les participants correspondent bien au profil des utilisateurs représentatifs.
- La liste des tâches à accomplir et les critères requis pour qu'une tâche soit considérée comme accomplie.
- Une liste de consignes aux observateurs, à rappeler avant le début du test.
- Des feuilles d'observation, pour noter les horaires, les actions des participants, les problèmes et les commentaires.
- Un questionnaire post-évaluation, pour mesurer la satisfaction et la compréhension de l'utilisateur et pour recueillir toute autre information que le participant voudrait livrer.

Nous considérons dans notre système 10 utilisateurs pour identifier la majorité des problèmes d'utilisabilité. En outre, la durée d'un test d'utilisabilité de notre système collaboratif a pris une quinzaine de minutes pour se familiariser avec les fonctionnalités de l'interface.

#### 5.2.5.3 Protocole Expérimental

En premier lieu, 10 utilisateurs se connectent sur le système en utilisant un nom d'utilisateur et un mot de passe expérimental. Ainsi, chaque utilisateur possède une liste de tâche basique à effectuer :

- Uploadé une image, un vidéo, et un fichier audio.
- Glisser/déposer ces fichiers sur la carte 2D. Le système attribue des coordonnées selon l'endroit où les utilisateurs les ont déposés.
- Dessiner un segment qui définit la trajectoire faite par un plongeur (utilisateur). Cette trajectoire est effectuée en définissant deux ou plusieurs points sur la carte, qui seront reliés automatiquement par le système. Ainsi, une trajectoire est une série de points avec des coordonnées (x,y).
- Lire un fichier vidéo, un fichier audio ou agrandir une image.
- Modifier les commentaires des fichiers déposés.

Après, ils effectuent des actions plus complexes :

- Composer des segments : Les utilisateurs qui collaborent entre eux, peuvent créer chacun un segment qui définit la trajectoire faite par chacun d'eux. Ceci permet de créer de multiples trajectoires qui représentent les expériences de plusieurs utilisateurs.
- Déposer des fichiers sur les trajectoires : Chaque utilisateur va tenter de déposer ses fichiers sur sa trajectoire. Ainsi, chacun des utilisateurs définit l'ensemble des images, vidéos ou fichiers audio prises sur sa trajectoire.
- Créer une surface qui représente une région dans la carte. Une composition de surfaces s'effectue à partir de plusieurs points de références reliés par des segments. Ainsi, les utilisateurs définissent leurs points, puis choisissent la couleur de la surface, qui définit la profondeur de celle-ci dans la région sous-marine représentée.
- Intégrer des services par la spécification de mots clés. Cette procédure est effectuée par le biais de moteur de recherche, qui est interfacé avec des UDDIs extérieurs au système.

Lors des tests de l'ergonomie de l'interface de *OceanydGroupware*, nous avons soumis un panel de testeurs ne connaissant pas le système. Les testeurs sont des doctorants dans notre laboratoire qui sont majoritairement des informaticiens. Chaque utilisateur a pu tester l'interface de collaboration sur sa propre machine au sein du laboratoire. Afin de s'assurer d'une certaine qualité d'ergonomie, un questionnaire a permis de recueillir les informations pertinentes. Lors de la manipulation, certains paramètres ont été fixés, comme l'environnement matériel et logiciel, les missions effectuées et le nombre de participants. Le protocole pour le sondage effectué est le suivant : les testeurs n'ayant jamais manipulé l'application, reçoivent une feuille d'instructions détaillant les étapes de l'expérience. Ils doivent collaborer pour effectuer les diverses tâches sur l'interface de *OceanydGroupware*. Après les manipulations, les utilisateurs ont répondus aux questionnaires, qui comprennent des questions liées respectivement à :

- L'utilisabilité : l'efficacité d'uploadé des fichiers, de dessiner des trajectoires, de créer des surfaces.
- La satisfaction : L'affichage correcte des fichiers sur la carte, la pertinence des explications sur l'interface.
- La collaboration : La visibilité des actions des utilisateurs, la composition de surfaces, l'intégration de services web.
- L'appréciation : Pertinence du mécanisme du chat, l'appréciation générale de l'utilisation, l'ergonomie de l'interface.

Une cinquième catégorie permet de récupérer les remarques diverses

#### 5.2.5.4 Interprétation des résultats

La première donnée que nous pouvons extraire est la moyenne globale des notes, qui est de 2.38/4, exprimant une qualité d'ergonomie plutôt appréciée. En comparant les catégories entre elles, nous remarquons que les moyennes d'utilisabilité et d'appréciation de l'interface ont eu les meilleures notes : le point le plus critiqué dans les commentaires apportés est la composition de surfaces, qui n'est apparemment pas bien visible dans l'interface. De point de vue de l'intégration de services (la partie "collaboration" de l'évaluation), les utilisateurs ont apprécié l'ajout de nouvelles fonctionnalités selon leurs besoins. En même temps, concernant la collaboration, les utilisateurs ayant été les plus critiques ont fait des reproches du point de vue de qualité de services intégrés, où ces services pour ce domaine d'application, ne sont pas considérés très pertinents pour rapport à la collaboration. Une synthèse est faite dans le tableau 5.1. Le graphique correspondant est donné dans la figure 5.22, qui présente les moyennes par aspect.

| Catégorie | Moyenne |
|---|---|
| I- Utisabilité | 0.76 |
| II- Satisfaction | 0.52 |
| III- Collaboration | 0.4 |
| IV- Appréciation | 0.7 |
| Moyenne | 2.38/4 |

Tab. 5.1 – Synthèse des moyennes des notes données, par catégorie.

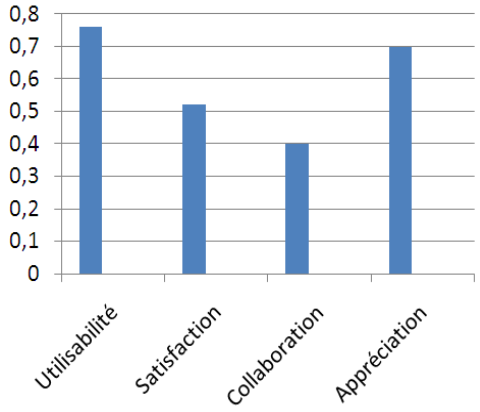



FIG. 5.22 – Représentation des moyennes pour chaque aspect dans l'évaluation subjective

Les principales remarques faites sont les suivantes :

- Support des vidéos pour un seul type d'extension (.flv)
- Problème d'encodage des caractères.
- Manque de clarté sur les manipulations possibles dans l'interface.
- Le rafraîchissement de la page est manuel afin de voir les actions des autres utilisateurs.
- Trop de services web disponibles pour un mot clés.
- Composition de surfaces un peu compliqué.
- Couleur des trajectoires pas suffisantes pour illustrer tout les profondeurs possibles.

## 5.3 Conclusion

Dans ce chapitre, nous avons présenté deux études de cas afin de mettre en œuvre les deux architectures logicielles $U3D$ et $UDDI4C$, dans deux domaines d'applications différents : la téléopération sur internet (projet ARITI), et la collecte de données multimédia (projet DIGITAL OCEAN) :

- Pour le projet ARITI, le but est de remédier les contraintes liées à l'usage limité de l'application par les missions statiques. Ainsi, nous avons fait évoluer le système de sorte que les utilisateurs puissent générer leurs propres missions dynamiquement, sans arrêter l'exécution du système, par l'application de l'architecture logicielle $U3D$.

- Pour le projet DIGITAL OCEAN, nous avons démontré l'intérêt de l'utilisation des services web dans le cadre de la collecte des données multimédias, afin que l'on puisse les exploiter d'une façon plus précise pour enrichir des environnements sous-marins en 3D, par l'application de l'architecture logicielle $UDDI4C$.

Ensuite, nous avons mis en œuvre une évaluation subjective pour l'interface de *OceanydGroupware*. Nous avons discuté la nécessité d'avoir des évaluations dans le processus du développement d'un système. Il s'est avéré que l'évaluation des architectures logicielles est une tâche assez complexe, surtout si les composants de l'architecture sont distribués sur internet tels les services web. Ainsi, les évaluateurs du système feront face à de nombreuses contraintes, tels des incertitudes de la disponibilité de ces composants logicielles.

Ces problèmes sont particulièrement précis lorsque les services qui sont utilisés par des êtres humains ou les agents logiciels, sont externes. Ils sont moins graves lorsque les services utilisés sont locales et ainsi le niveau de confiance sera plus grand. Dans ces cas, le code source peut être disponible afin de guider le processus de test. La résolution de ces problèmes de test en produisant des orientations, des outils et des techniques de test pour des applications orientées service est actuellement un important domaine de recherche.

# CONCLUSION ET PERSPECTIVES

## 5.4 Résumé de la thèse et contributions

Dans cette thèse, nous avons proposé deux modèles d'architecture logicielle pour mettre en œuvre la malléabilité des collecticiels. Ainsi, l'objectif de notre travail a été de modéliser, de concevoir et de mettre en œuvre un système de collaboration malléable, capable de prendre en charge la collaboration de plusieurs utilisateurs pour les aider à réaliser des tâches communes, tout en améliorant le processus de la collaboration. A partir de notre état de l'art, nous avons constaté que les systèmes collaboratifs qui existent, ne prennent pas en compte explicitement la propriété de la malléabilité. Ces collecticiels sont basés sur des composants rigides qui, éventuellement, limitent la collaboration entre les individus. Nous avons essayé de prendre en compte les propriétés de la malléabilité, en proposant une solution basée sur un formalisme multi-agent combiné avec un formalisme basé sur les services web, afin de créer une architecture logicielle malléable et générique pour les collecticiels. Nous résumons ci-dessous les principales contributions dans chaque chapitre de la thèse :

Dans le premier chapitre, nous avons étudié les modèles architectures logicielles des collecticiels qui se trouvent dans la littérature, ainsi que les collecticiels qui dérivent de ces modèles. Nous avons effectué une classification et analyse de ces travaux par rapport à des propriétés telles que l'ouverture, l'adaptabilité et l'interopérabilité des systèmes. Nous avons remarqué que la plupart de ces modèles sont fermés, ce qui veut dire que la génération de nouvelles fonctionnalités est, soit limité à quelques services internes, soit quasiment impossible.

Dans le deuxième chapitre, nous avons étudié les technologies les plus répandues dans les constructions des logiciels collaboratifs et interopérables, notamment les systèmes multi-agents (SMA) et les services web. Nous avons discuté les avantages et les propriétés de chaque technologie. Ensuite, nous avons présenté les travaux réalisés dans le domaine de l'intégration des ces deux technologies. Il s'est avéré que leurs intégrations offrent de nombreuses propriétés pour pallier leurs faiblesses individuelles. Ainsi, nous avons analysé les avantages de leurs intégrations dans le domaine de la conception des architectures de collaboration. Cette étude a montrée que l'intégration de ces deux technologies peuvent apporter de nombreuses solutions pour remédier au problème de la malléabilité des collecticiels, qui est notre problématique principale.

Dans le troisième chapitre, nous avons proposé un nouveau formalisme pour la conception des collecticiels malléables. Dans la première section, nous avons proposé un formalisme basé sur les services web pour construire des logiciels collaboratifs. Ensuite, nous

avons décrit l'architecture logicielle associée à ce formalisme, que nous appelons $UDDI4C$. Dans la deuxième section, nous avons introduit un formalisme à bases de services web et des agents logiciels. Dans ce formalisme, un agent WAG ou Web AGent, est un agent hybride, capable de rechercher de nouvelles ressources sur internet, et les consommer en tant que nouveaux comportements. Finalement, nous avons présenté le modèle d'architecture associé à ce formalisme, l'$U3D$.

Le quatrième chapitre est découpé en deux sections. Tout d'abord, nous avons abordé les technologies utilisées pour mettre en œuvre notre architecture logicielle $U3D$. Ensuite, nous avons décrit une étude en UML pour modéliser les différentes classes du système. Nous avons présenté les mécanismes de traductions de messages entre les environnements de services web et des agents par un composant principal de notre architecture : le WSIG (Web Service Integration Gateway). Nous avons proposé des algorithmes qui visent à mettre en œuvre la malléabilité du système via l'intégration et la composition de services. Dans une seconde partie, nous avons proposé une étude de performances des deux architectures logicielles proposées en termes d'émissions de messages. Nous avons appliqué l'architecture $UDDI4C$ comme un support complémentaire pour introduire la malléabilité dans le SMA-C (Khezami, 2005). Ainsi, nous avons pu calculer le coût supplémentaire que va engendrer la malléabilité dans le système. Ensuite, nous avons proposé une étude de l'architecture logicielle $U3D$ (en remplacement du SMA-C), et nous avons comparé les performances des deux modèles d'architecture logicielle. Cela nous conduit à la conclusion que notre architecture $U3D$ offre de meilleures performances en termes d'émissions de messages tout en assurant la malléabilité des systèmes collaboratifs.

Dans le cinquième chapitre, nous avons présenté deux études de cas afin expliquant l'apport des deux architectures logicielles pour introduire la malléabilité dans deux projets différents : le projet ARITI et le projet DIGITAL OCEAN. Dans le projet ARITI, nous avons montré comment l'architecture $U3D$ peut apporter de la malléabilité grâce au processus de composition de missions à la volée, afin de générer de nouveaux comportements du robot. Ceci est effectué par l'accès à une ontologie qui décrit les manipulations éventuelles du robot, afin de générer du code de la nouvelle mission composée par les utilisateurs. La génération automatique de code est effectuée en utilisant le plugin BeanOntologyGenerator de la plateforme Protégé. Dans le projet DIGITAL OCEAN, nous avons montré comment l'architecture $UDDI4C$ peut apporter de la malléabilité grâce au processus d'intégration et de composition de services. Il s'est avéré que l'utilisation des services web pourrait avoir une utilité afin d'obtenir de nouvelles informations sur les fichiers collectés par les utilisateurs, ainsi que pour améliorer le processus de la collaboration.

Les travaux qui ont été accomplis jusqu'à présent nous ont permis de construire un ensemble cohérent de modèles d'architecture logicielle pour introduire de la malléabilité des collecticiels. Sur le plan de l'implémentation, le choix des technologies de développment Web (services web) favorise l'ouverture des architectures proposées à d'autres applications ou services sur le Web. Cette approche constitue un axe de développement prometteur pour l'exploitation des applications collaboratives.

## 5.5 Limites et perspectives envisagées

Il s'est avéré que l'intégration de ces deux technologies (agent et service web) produit une synergie qui amène une certaine ouverture du système. Celle ci est une propriété importante pour qu'un collecticiel puisse évoluer au cours du temps, et s'adapter aux préférences des utilisateurs face à leurs besoins émergeants. Vu le nombre émergeant des services web sur internet, notre système actuel ne permet pas la classification de ces services. Ainsi, il se contente de proposer le premier service web trouvé qui peut satisfaire les besoins des utilisateurs. Pour les travaux envisagés dans le futur, nous proposons des perspectives sur un court, moyen et long terme :

- Sur le court terme, notre premier but est de mettre l'utilisateur au centre de la collaboration malléable. Ainsi, nous mettons en œuvre une collaboration homme-homme malléable. Cette collaboration constitue une grande évolution dans la malléabilité des collecticiels, où dans ce contexte (TCAO) et pour les chercheurs dans ce domaine, l'utilisateur est toujours l'acteur majeur et le premier a être influencer par la nature de la collaboration. Un deuxième but serait de proposer des services adaptés aux performances des machines et le réseau internet. En conséquence, nous tentons de mettre l'accent sur les attributs non-fonctionnels des services (QoS), qui constituera un atout pour mesurer le degré de la malléabilité d'un système collaboratif. Nous avons commencé à exploiter ce domaine en proposant une interface pour la recherche des services web à partir de leurs QoS (Annexes B).

- Sur le moyen terme, le travail envisagé est d'exploiter l'architecture $U3D$ pour le télétravail collaboratif autour des environnements de Réalité Virtuelle (RV) et de Réalité Augmentée (RA). Il s'agit d'adapter l'$U3D$ pour la conception et le développement de nouvelles architectures logicielles de collaboration, adaptées aux nouvelles IHM multi sensorielles (nouvelles modalités de perception d'interaction et de communication), offertes par les environnements de RV/RA. Ces applications s'inscrivent dans le cadre du télétravail collaboratif en réalité mixte autour de la plateforme EVR@[1] de notre laboratoire.

- Sur le long terme, notre but serait de générer des ontologies dynamiquement, selon les préférences des utilisateurs. Ainsi, le système pourra s'adapter pus facilement aux besoins des futures usagées, qui tentent de réutiliser des services déjà proposés pour une tâche similaire ou identique. Nous visons également de chercher d'autres mécanismes pour la malléabilité des collecticiels basés sur des agents logiciels, pour augmenter leurs capacités décisionnelles.

Ainsi, de nombreuses questions sont posées, telles que : comment réaliser efficacement l'automatisation du web, la composition et l'invocation ? Comment prendre en charge la gestion d'exécution des services web par les agents logiciels afin d'améliorer le processus de la collaboration ? Toutes ces questions ouvrent d'importantes pistes de recherche à explorer. Nous continuons nos recherches dans le domaine du TCAO en utilisant les services web et les agents logiciels, ainsi que le web sémantique qui fait le lien entre ces deux technologies.

[1] http ://evra.ibisc.univ-evry.fr

# Bibliographie


Akkiraju, R., Goodwin, R., Doshi, P., and Roeder, S. (2003). A method for semantically enhancing the service discovery capabilities of UDDI. In *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 87–92.

Alda, S., Kuck, J., and Cremers, A. (2007). Tailorability of personalized BPEL-based Workflow Compositions. In *The 2007 IEEE Congress on Services*, pages 245–252.

Anderson, G., Graham, T., and Wright, T. (2000). Dragonfly : linking conceptual and implementation architectures of multiuser interactive systems. In *Proceedings of the 22nd international conference on Software engineering*, pages 252–261. ACM New York, NY, USA.

Bass, L., Faneuf, R., Little, R., Mayer, N., Pellegrino, B., Reed, S., Seacord, R., Sheppard, S., and Szczur, M. (1992). A metamodel for the runtime architecture of an interactive system. *SIGCHI Bulletin*, 24(1) :32–37.

Biemans, M. and Ter Hofte, G. (1999). Tailorability : state-of-the-art. *Gigaport project deliverable, Telematica Instituut, The Netherlands*.

Blois, M., Escobar, M., and Choren, R. (2007). Using agents and ontologies for application development on the semantic web. *Journal of the Brazilian Computer Society*, 13 :35–44.

Bourguin, G. (2000). *Un support informatique à l'activité coopérative fondé sur la Théorie de l'Activité : le projet DARE*. PhD thesis, University of Sciences and Technology of Lille.

Bourguin, G. (2003). Lessons learned from the implementation of a reflexive groupware system. In *Proceedings of the 15th French-speaking conference on human-computer interaction on 15eme Conference Francophone sur l'Interaction Homme-Machine*, pages 40–47. ACM New York, NY, USA.

Bourguin, G. (2004). Proposition pour une gestion dynamique de l'inter-activités dans le TCAO. In *Proceedings of the 16th conference on Association Francophone d'Interaction Homme-Machine table of contents*, pages 191–194. ACM New York, NY, USA.

Buhler, P. and Vidal, J. (2002). Toward the synthesis of web services and agent behaviors. In *Proceedings of the Agentcities : Challenges in Open Agent Environments Workshop*, pages 25–31.

Buhler, P. and Vidal, J. (2003). Semantic web services as agent behaviors. *Agentcities : Challenges in Open Agent Environments*, pages 25–31.

Cabri, G., Ferrari, L., and Leonardi, L. (2003). A case study in role-based agent interactions. *12th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 03)*.

Cabri, G., Ferrari, L., and Leonardi, L. (2004). Towards the use of mobile agent based message systems. *13th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 04)*, pages 27–32.

Calvary, G., Coutaz, J., and Nigay, L. (1997). From single-user architectural design to PAC* : a generic software architecture model for CSCW. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 242–249. ACM New York, NY, USA.

Cheaib, N., Otmane, S., and Mallem, M. (2008a). Combining fipa agents and web services for the design of tailorable groupware architecture. In *Proc ACM 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS2008), ACM SIGWEB*, pages 702–705, ACM New York, NY, USA.

Cheaib, N., Otmane, S., and Mallem, M. (2008b). Integrating internet technologies in designing a tailorable architecture. In *Proceedings of the 12th international conference on Computer Supported Cooperative Work (CSCWD)*, pages 141–147, Xi'An, China.

Cheaib, N., Otmane, S., and Mallem, M. (2009). Collaborative multimedia collection for enriching and visualizing 3d underwater sites. In *6th IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, United-States*, ACM New York, NY, USA.

Cheaib, N., Otmane, S., and Mallem, M. (2010). Web services and software agents for tailorable groupware design. In *Emergent Web Intelligence : Advanced Semantic Technologies*, page to appear. Springer Book.

Cheaib, N., Otmane, S., Mallem, M., Dinis, A., and Fiès, N. (2008c). Oceanyd : a new tailorable groupware for digital media collection for underwater virtual environments. In *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts, ACM SIGGRAPH*, pages 256–263, ACM New York, NY, USA.

Dangelmaier, W., Hamoudia, H., and Klahold, R. (2002). Domain Preferences for End-User Tailoring in Shared Workflow Interfaces. In *CRIWG Workshop, Darmstadt, Germany*.

Dewan, P. (1999). Architectures for collaborative applications. *Journal of Computer Supported Co-operative Work*, 7 :169–193.

Dickinson, I. and Wooldridge, M. (2005). Agents are not (just) web services : considering BDI agents and web services. In *Proceedings of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE'2005), Utrecht, The Netherlands*.

Dillenbourg, P., Baker, M., Blaye, A., and O'Malley, C. (1996). The evolution of research on collaborative learning. *Learning in Humans and Machine : Towards an interdisciplinary learning science*, pages 189–211.

Dinis, A., Fies, N., Cheaib, N., Otmane, S., Mallem, M., Nisan, A., and Boi, J. (2008). Digital ocean : A national project for the creation and distribution of multimedia content for underwater sites. In *Proc 14th International Conference on Virtual Systems and Multimedia VSMM'08, Limassol, Cyprus*, pages 389–396.

Dutra, M., Ghodous, P., Kuhn, O., and Tri, N. (2010). A Generic and Synchronous Ontology-based Architecture for Collaborative Design. *Concurrent Engineering, Research and Applications*, 18(1) :65–74.

Ellis, A. (1994). Conceptual model of groupware. *Proc of the International Conference on Computer Supported Cooperative Work (CSCW), ACM Press NY*, pages 79–88.

Ferber, J. (1995). Les systèmes multi-agents. *Vers une intelligence collective. InterEditions.*

Fernandez, A. (2005). *Groupware for Collaborative Tailoring.* PhD thesis, University of Hagen - Germany.

Foukarakis, I., Kostaridis, A., Biniaris, C., Kaklamani, D., and Venieris, I. (2007). Webmages : An agent platform based on web services. *Computer Communications*, 30(3) :538–545.

Frey, D., Stockheim, T., Woelk, P., and Zimmermann, R. (2003). Integrated multi-agent-based supply chain management. *12th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 03).*

Grundy, J. and Hosking, J. (2002). Developing adaptable user interfaces for component-based systems. *Interacting with Computers*, 14(3) :175–194.

Guest, S. (2005). Top 10 tips for web services interoperability, http ://msdn.microsoft.com/msdntv/episode.aspx ?xml=episodes0210webservicessg.

Hill, R., Brinck, T., Rohall, S., Patterson, J., and Wilner, W. (1994). The rendezvous architecture and language for constructing multiuser applications. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 1(2) :81–125.

Huhns, M. and Singh, M. (2005). Service-oriented computing : Key concepts and principles. *IEEE Internet Computing*, 9(1) :75–81.

Jerstad, I., Dustdar, S., and Thanh, D. (2005). A service oriented architecture framework for collaborative services. *14th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprise, 2005*, pages 121–125.

Kahler, H. (2001). *Supporting collaborative tailoring.* PhD thesis, Department of Communication, Journalism and Computer Science. Roskilde University, Denmark.

Kaplan, S. and Carroll, A. (1992). Supporting collaborative processes with Conversation Builder. *Journal of Computer communications*, 15(8) :489–501.

Khezami, N. (2005). *Vers un collecticiel basé sur un formalisme multi-agent destiné à la téléoperation collaborative via Internet.* PhD thesis, IBISC (Informatique, Biologie Intégrative et Système Complexes) Laboratory- University of Evry Val d'Essonne - Evry, France.

Khezami, N., Otmane, S., and Mallem, M. (2005). A new formal model of collaboration by multi-agent systems. *Proc IEEE KIMAS : 32-37, Massachusetts, USA.*

Klöckner, K., Mambrey, P., Sohlenkamp, M., Prinz, W., Fuchs, L., Kolvenbach, S., Pankoke-Babatz, U., and Syri, A. (1995). Bridging the Gap between Bonn and Berlin for and with the Users. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work*, page 17. Springer.

Krasner, G. and Pope, S. (1988). A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *Journal of Object-Oriented Programming*, 1(3) :26–49.

Lamparter, S. and Ankolekar, A. (2006). Automated selection of configurable web services. *Journal of Int. Tagung Wirtschaftsinformatik*.

Laurillau, Y. (2002). *Conception et réalisation logicielles pour les collecticiels centrées sur l'activité de groupe : le modèle et la plate-forme Clover*. PhD thesis, University of Joseph Fourier, Grenoble, France.

Laurillau, Y. and Nigay, L. (2002a). Clover architecture for groupware. *Proc of the International Conference on Computer Supported Cooperative Work (CSCW), ACM Press*, pages 236–245.

Laurillau, Y. and Nigay, L. (2002b). Le modèle d'architecture Clover pour les collecticiels. In *Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine, IHM2002), ACM*, pages 113–120.

Lee, E. and Lee, B. (2007). An Agent-Based Web Service Composition Using Semantic Information and QoS. *Lecture Notes in Computer Science*, 4496 :928–937.

Maamar, Z., Sheng, Q. Z., and Benatallah, B. (2003). Interleaving web services composition and execution using software agents and delegation. *AAMAS Workshop*.

Maeda, C., Lee, A., Murphy, G., and Kiczales, G. (1997). Open implementation analysis and design. In *ACM SIGSOFT Software Engineering Notes*, pages 44–52.

Marsic, I. (1999). DISCIPLE : a framework for multimodal collaboration in heterogeneous environments. *ACM Computing Surveys (CSUR)*, 31(2es).

Matskin, M., Kungas, P., Rao, J., Sampson, J., and Petersen, S. (2005). Enabling web services composition with software agents. *Proceedings of the Ninth IASTED International Conference on Internet and Multimedia Systems and Applications, IMSA 2005, Honolulu, Hawaii, USA*, pages 93–98.

Menasce, D. (2002). QoS issues in Web services. *Journal of Internet Computing IEEE*, pages 72–75.

Morch, A. (1997). Three levels of end-user tailoring : customization, integration, and extension. *Journal in Computers and design in context*, pages 51–76.

Morch, A., Stiemerling, O., and Wulf, V. (1998). Tailorable groupware : issues, methods, and architectures. *ACM SIGCHI Bulletin*, 30(2) :40–42.

Nandigam, J., Gudivada, V., and Kalavala, M. (2005). Semantic web services. *Journal of Computing Sciences in Colleges*, 21(1) :50–63.

Neward, T. (2003). Web + services : http ://soa.sys-con.com/node/39848.

Nguyen, T. X. and Kowalczyk, R. (2005). Ws2jade : Integrating web service with jade agents. *Technical Report, SOCAB05*.

Nigay, L. (1994). *Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales*. PhD thesis, University of Joseph Fourier - Grenoble 1.

Obrst, L. (2003). Ontologies for semantically interoperable systems. In *Proceedings of the twelfth international conference on Information and knowledge management*, pages 366–369. ACM New York, NY, USA.

Otmane, S. (2000). *Télétravail Robotisé et Réalité Augmentée : Application à la Téléoperation via Internet*. PhD thesis, University of d'Evry Val-Essonne.

Otmane, S., Cheaib, N., and Mallem, M. (2008). *Internet-based Collaborative Teleoperation : Towards Tailorable Groupware for Teleoperation.*, chapter 7, pages 163–193. Wiley and ISTE/Hermes.

Otmane, S., Mallem, M., Kheddar, A., and Chavand, F. (2000). Ariti : an augmented reality interface for teleoperation on the internet. *Advanced Simulation Technologies Conference (ASTC 00) - High Performance Computing.* pages 254–261, Wyndham City Center Hotel, Washington, D.C., USA.

Palathingal, P. and Chandra, S. (2004). Agent approach for service discovery and utilization. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, page 9.

Pathak, J., Koul, N., Caragea, D., and Honavar, V. (2005). A framework for semantic web services discovery. In *Proceedings of the 7th annual ACM international workshop on Web information and data management*, pages 45–50. ACM New York, NY, USA.

Patterson, J. (1995). A taxonomy of architectures for synchronous groupware applications. *Journal of ACM SIGOIS Bulletin*, 15(3) :27–29.

Payet, D. (2003). *L'enrichissement de message comme support pour la composition logicielle.* PhD thesis, Unviversity of Montpellier 2.

Peltz, C. (2003a). Web services orchestration. a review of emerging technologies, tools and standards. pages 46–52. Hewlett Packard White Paper, IEEE Computer Society.

Peltz, C. (2003b). Web services orchestration and choreography. *Journal of Computer*, 36(10) :46–52.

Roseman, M. and Greenberg, S. (1997). Simplifying component development in an integrated groupware environment. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 65–72.

Sabou, M. (2006). *Building web service ontologies.* PhD thesis, SIKS- Dutch Graduate School and Knowledge Systems, University of VRIJE.

Salber, D. (1995). *De l'interaction individuelle aux systèmes multiutilisateurs. L'exemple de la Communication Homme-Homme- Médiatisée.* PhD thesis, Thèse de doctorat Informatique, University of Joseph Fourier, Grenoble, France.

Selliah, S., Reddy, R., Yu, J., Bharadwaj, V., and Reddy, S. (2004). Eksarva : A framework for enabling agent-based collaboration. *13th IEEE International Workshops on Enabling Technologies : Infrastructure for Collaborative Enterprises (WETICE 04)*, pages 33–38.

Slagter, R. (2004). *Dynamic Groupware Services : Modular design of tailorable groupware.* PhD thesis, Telematica Instituut, the Netherlands.

Slagter, R., Biemans, M., and Ter Hofte, H. (2001). Evolution in use of groupware : facilitating tailoring to the extreme. In *Proceedings. Seventh International Workshop on Groupware*, pages 68–73.

Slagter, R., Ter Hofte, G., and Stiemerling, O. (2000). Component-based groupware : An introduction. In *Proceedings of CBG2000, the CSCW2000 workshop on Component-Based Groupware)*, volume 2, pages 394–400.

Slagter, R., Ter Hofte, H., and Kruse, H. (2002). CoCoWare .NET architecture. *Telematica Instituut report series (TI/RS/2001/091), Enschede, The Netherlands*, 1.

Starr, B., Ackerman, M., and Pazzani, M. (1996). Do-I-Care : A collaborative web agent. In *Conference on Human Factors in Computing Systems*, pages 273–274. ACM New York, NY, USA.

Stefik, M., Bobrow, D., Foster, G., Lanning, S., and Tatar, D. (1987). WYSIWIS revised : early experiences with multiuser interfaces. *ACM Transactions on Information Systems (TOIS)*, 5(2) :147–167.

Stiemerling, O. (1997a). CAT : Component architecture for tailorability. *Department of Computer Science, University of Bonn, Germany, Working Paper*.

Stiemerling, O. (1997b). Supporting tailorability of groupware through component architectures. *The International Workshop on Object Oriented Groupware Platforms, OOGP'97*, 7 :54–60.

Stiemerling, O. and Cremers, A. (1998). Tailorable component architectures for cscw-systems. In *Proceedings of the 6th Euromicro Workshop on Parallel and Distributed Programming*, pages 21–24.

Stiemerling, O., Hinken, R., and Cremers, A. (1999). Distributed component-based tailorability for CSCW applications. In *Autonomous Decentralized Systems, 1999. Integration of Heterogeneous Systems. Proceedings. The Fourth International Symposium on*, pages 345–352.

Svirskas, A., Wilson, M., Matthews, B., Arenas, A., Mac Randal, D., Gallop, J., Bicarregui, J., and Lambert, S. (2005). Towards an Efficient, Reliable and Collaborative Web : from Distributed Computing to Semantic Description, Composition and Matchmaking of Services. In *W3C workshop on Frameworks for Semantics in Web Services*.

Syri, A. (1997). Tailoring cooperation support through mediators. In *Proceedings of 5th European Conference on Computer Supported Cooperative Work, Lancaster, UK*, pages 157–172.

Tarpin-Bernard, F. and David, B. (1997). Amf : A new design pattern for complex interactive software ? *Advances in Human Factors Ergonomics*, 21 :351–354.

Teege, G. (1999). A CSCW System Model for Classifying Tailorability Implementations. In *Proc. of Workshop on Implementing Tailorability in Groupware*.

Teege, G. (2000). Users as composers : Parts and features as a basis for tailorability in CSCW systems. *Computer Supported Cooperative Work (CSCW)*, 9(1) :101–122.

Ter Hofte, G. (1998). Working apart together : Foundations for component groupware. *Telematica Instituut Fundamental Research Series*, 1.

Ter Hofte, G., van der Lugt, H., and Houtsma, M. (1996). Co 4, a comprehensive model for groupware functionality. In *Telematics in a multimedia environment : Proceedings of Euromedia*, volume 96, pages 19–21.

Torres, D., Fernandez, A., Rossi, G., and Gordillo, S. (2007). Fostering Groupware Tailorability Through Separation of Concerns. *Lecture Notes in Computer Science*, 4715 :143.

Varga, L. and Hajnal, A. (2003). Engineering web service invocations from agent systems. *Lecture Notes in Computer Science*, pages 626–635.

Won, M., Stiemerling, O., and Wulf, V. (2006). Component-based approaches to tailorable systems. *End-User Development*, pages 115–141.

Wright, M. (2005). A detailed investigation of interoperability for web services. Master Thesis, Rhodes University, South Africa.

Wulf, V. (1999). Let's see your search tool ! collaborative use of tailored artifacts in groupware. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 50–59. ACM New York, NY, USA.

Wulf, V. and Golombek, B. (2001). Direct activation : A concept to encourage tailoring activities. *Behaviour & Information Technology*, 20(4) :249–263.

Wulf, V., Pipek, V., and Won, M. (2007). Component-based tailorability : Enabling highly flexible software applications. *International Journal of Human-Computer Studies*.

# Troisième partie

# Annexes

# Annexe A

## A.1 Données de l'étude de l'UDDI4C et l'U3D

### A.1.1 Données de l'UDDI4C

Les données attendues sont les messages émis par chaque agent durant chaque étape, avant et après l'application de $UDDI4C$ sur $SMA-C$. L'échantillon de test est composé de 41 essais, avec des nombres d'agents variant de 2 à 35. Un extrait des données est donné dans les tableaux A.1, A.2 et A.3.

| NbAgents | NbMCom-SMA-C | NbMCom-UDDI4C$_I$ | NbMCom-UDDI4C$_C$ |
|---|---|---|---|
| 2 | 10 | 16 | 14 |
| 4 | 22 | 34 | 30 |
| 6 | 34 | 52 | 46 |
| 10 | 58 | 88 | 78 |
| 20 | 118 | 178 | 158 |
| 35 | 208 | 313 | 278 |

Tab. A.1 – Echantillon des jeux de tests effectués, pour la partie "communication". NbM correspond au nombre de messages émis.

| NbAgents | NbMCoo-SMA-C | NbMCoo-UDDI4C$_I$ | NbMCoo-UDDI4C$_C$ |
|---|---|---|---|
| 2 | 32 | 40 | 36 |
| 4 | 80 | 96 | 88 |
| 6 | 128 | 152 | 140 |
| 10 | 224 | 264 | 242 |
| 20 | 560 | 620 | 580 |
| 35 | 824 | 964 | 894 |

Tab. A.2 – Echantillon des jeux de tests effectués, pour la partie "coordination". NbM correspond au nombre de messages émis.

| NbAgents | NbMPro-SMA-C | NbMPro-UDDI4C$_I$ | NbMPro-UDDI4C$_C$ |
|---|---|---|---|
| 2 | 47 | 10 | 10 |
| 4 | 103 | 14 | 14 |
| 6 | 159 | 18 | 18 |
| 10 | 271 | 26 | 26 |
| 20 | 551 | 46 | 46 |
| 35 | 971 | 76 | 76 |

TAB. A.3 – Echantillon des jeux de tests effectués, pour la partie "production". NbM correspond au nombre de messages émis.

### A.1.2 Données de l'U3D

Les données attendues sont les messages émis par chaque agent durant chaque étape. L'échantillon de test est composé de 41 essais, avec des nombres d'agents variant de 2 à 35. Un extrait des données récupérées est donné dans les tableaux A.4, A.5 et A.6.

| NbAgents | NbMComm-SMA-C | NbMCom$_I$ | NbMCom$_C$ |
|---|---|---|---|
| 2 | 10 | 5 | 6 |
| 4 | 22 | 9 | 12 |
| 6 | 34 | 13 | 18 |
| 10 | 58 | 21 | 30 |
| 20 | 118 | 41 | 60 |
| 35 | 208 | 71 | 105 |

TAB. A.4 – Echantillon des jeux de tests effectués, pour la partie "communication". NbM correspond au nombre de messages émis.

| NbAgents | NbMCoo-SMA-C | NbMCoo$_I$ | NbMCoo$_C$ |
|---|---|---|---|
| 2 | 32 | 5 | 4 |
| 4 | 80 | 9 | 8 |
| 6 | 128 | 13 | 12 |
| 10 | 224 | 21 | 20 |
| 20 | 560 | 41 | 40 |
| 35 | 824 | 71 | 70 |

TAB. A.5 – Echantillon des jeux de tests effectués, pour la partie "coordination". NbM correspond au nombre de messages émis.

| NbAgents | NbMPro-SMA-C | NbMPro$_I$ | NbMPro |
|---|---|---|---|
| 2 | 47 | 8 | 8 |
| 4 | 103 | 18 | 16 |
| 6 | 159 | 28 | 24 |
| 10 | 271 | 48 | 40 |
| 20 | 551 | 98 | 80 |
| 35 | 971 | 173 | 140 |

TAB. A.6 – Echantillon des jeux de tests effectués, pour la partie "production". NbM correspond au nombre de messages émis.

Dans le tableau A.7, nous comparons le nombre de messages totales (dans toutes les phases de collaboration) du système SMA-C avec notre système dans les deux processus d'orchestration (intégration et composition). Nous remarquons que notre système émet moins de messages que l'ancienne version. Les résultats sont exploités dans le graphe 4.30.

| NbAgents | NbM-SMA-C | NbM-U3D$_I$ | NbM-U3D$_C$ |
|---|---|---|---|
| 2 | 48 | 17 | 18 |
| 4 | 108 | 35 | 36 |
| 6 | 168 | 53 | 54 |
| 10 | 288 | 89 | 90 |
| 20 | 684 | 179 | 180 |
| 35 | 1048 | 314 | 315 |

TAB. A.7 – Nombres de messages totaux dans la collaboration dans SMA-C, et notre architecture U3D dans les deux mécanismes d'orchestration. NbM correspond au nombre totale de messages émis.

### A.1.3 Degré de malléabilité d'un collecticiel

Ce modèle fonctionnel permet d'introduire la malléabilité dans les trois espaces du collecticiel, qui sont la communication, la coordination et la production. Nous définissons le degré de la malléabilité d'un collecticiel, comme la capacité de ce dernier à composer et à intégrer des services, suivant les trois espaces fonctionnelles du collecticiel. Ainsi, plus les trois espaces peuvent subir des mécanismes de composition et d'intégration, plus la collaboration est considérée comme malléable. Nous pouvons voir une illustration dans la figure A.1.

A partir de la définition de la malléabilité des collecticiels, nous définissons le $Degré(C)$ et $Degré(I)$ comme étant le degré de malléabilité par rapport à la Composition (C), et l'intégration (I). Par exemple, $Degré(C)_{Com}$ est le degré de composition dans l'espace de communication, et $Degré(I)_{Com}$ est le degré de l'intégration dans l'espace de communication. Ce degré s'applique sur tous les espaces du trèfle fonctionnel.

Soit $DM$ la fonction qui définit le degré de malléabilité :

$$DM_{Coll} = f(DM_{Comm}, DM_{Coor}, DM_{Prod}) = [0 - 1]$$

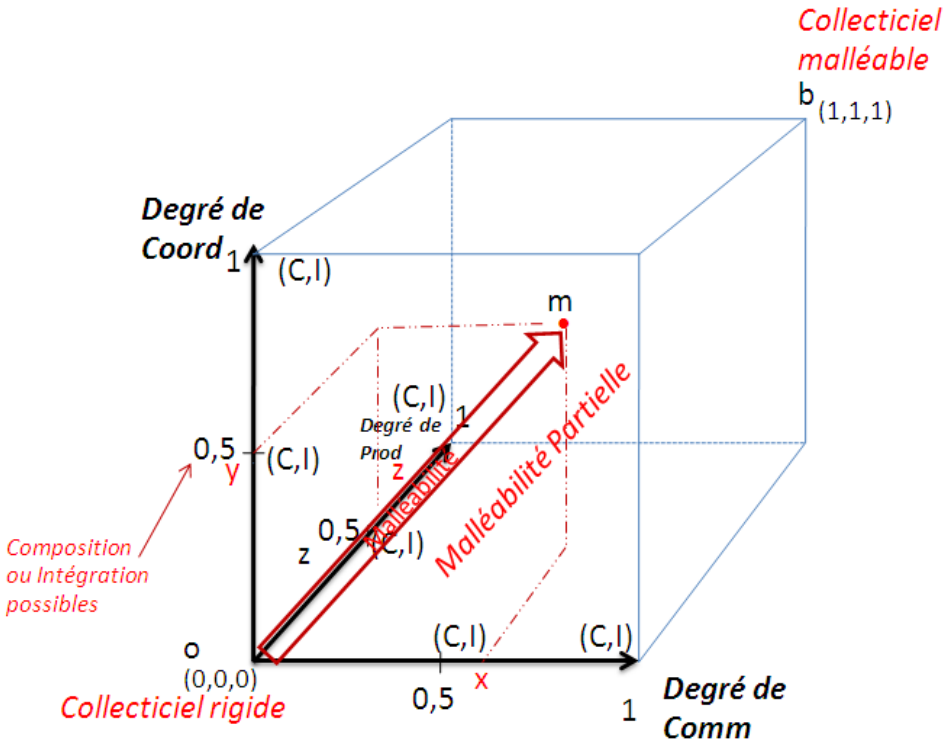



FIG. A.1 – Degré de malléabilité dans les collecticiels

Ainsi, $DM$ est la fonction des degrés de malléabilité de chaque espace du trèfle fonctionnel des collecticiels, avec :

$$DM_{Comm} = Degré(C) + Degré(I) \in \{0, 0.5, 1\}$$
$$DM_{Coor} = Degré(C) + Degré(I) \in \{0, 0.5, 1\}$$
$$DM_{Prod} = Degré(C) + Degré(I) \in \{0, 0.5, 1\}$$

En conséquence, dans chaque espace de collaboration, il peut y avoir les degrés de malléabilité suivants :

- Si les deux mécanismes de l'intégration et de la composition s'appliquent, nous attribuons un degré de malléabilité de 1.
- Si un des deux mécanismes s'applique, nous attribuons un degré de malléabilité de 0.5.
- Si aucun de deux mécanismes s'appliquent, nous attribuons un degré de malléabilité de 0.

Prenons par exemple, un point $m$ de l'espace. Nous définissons la granularité de malléabilité du vecteur $\overrightarrow{om}$, comme étant :

$$DM_{Coll} = \sqrt{x^2 + y^2 + z^2}/\sqrt{3}$$
$$\Rightarrow \|\overrightarrow{om}\|/\sqrt{3}$$

Sachant que $o = (0, 0, 0)$ représente l'origine de l'espace.

Ainsi, nous pouvons définir la granularité de la malléabilité des collecticiels comme étant la norme du vecteur $\overrightarrow{ob}$ égale à 1, telle que $b = (1, 1, 1)$, et qui représente le point optimale de la malléabilité.

*Remarque sur le degré de malléabilité :*
*Le calcul du degré de malléabilité représente uniquement un indicateur quantitatif sur la capacité d'un collecticiel à composer et/ou à intégrer des services, l'aspect qualitatif n'est pas pris en compte dans le modèle proposé.*

# Annexe B

## B.1 Conception de *OceanydGroupware*

### B.1.1 Diagrammes UML

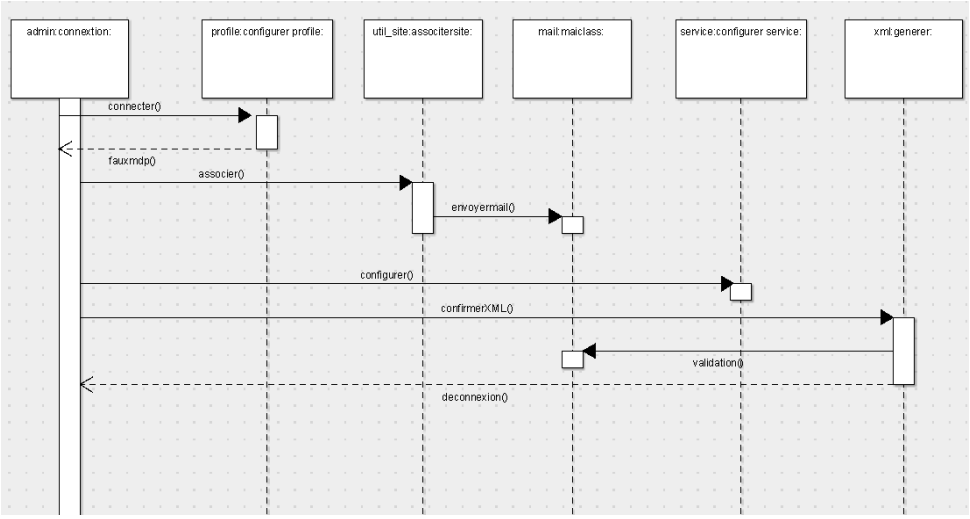



Fig. B.1 – Administrateur - Diagramme de séquence

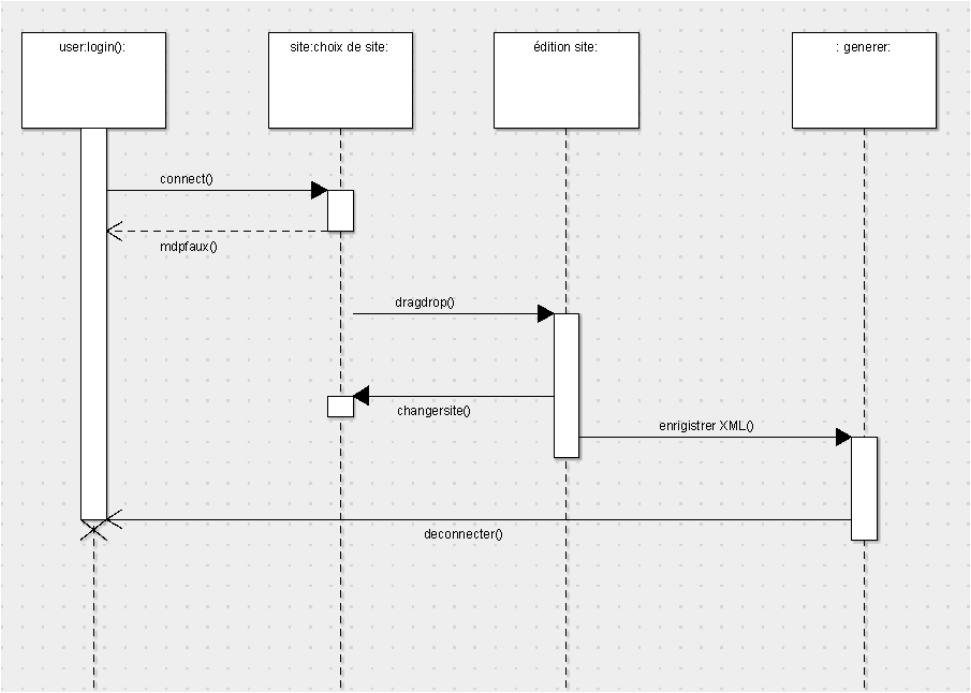



Fig. B.2 – Utilisateur - Choix de sites

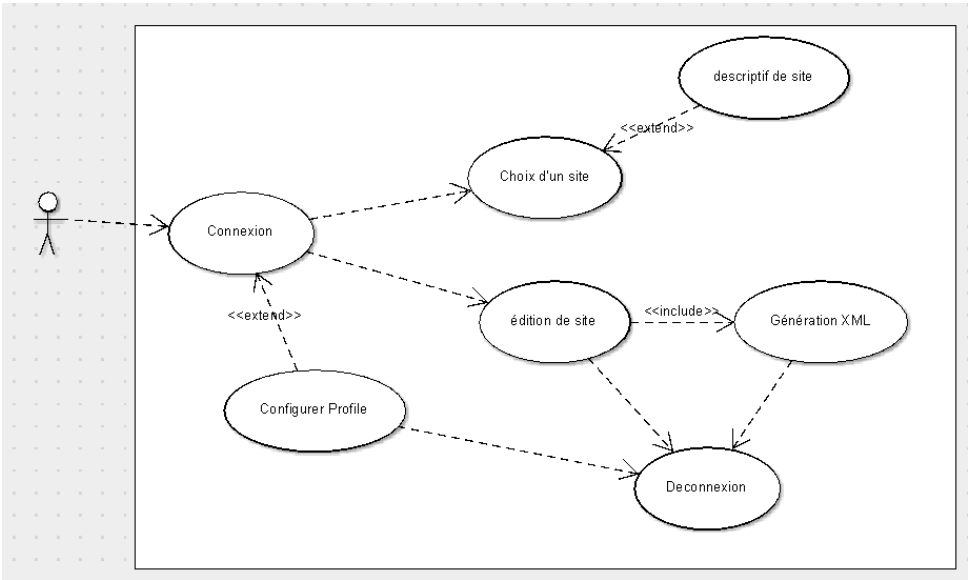



Fig. B.3 – Diagramme de cas d'utilisation - Utilisateur

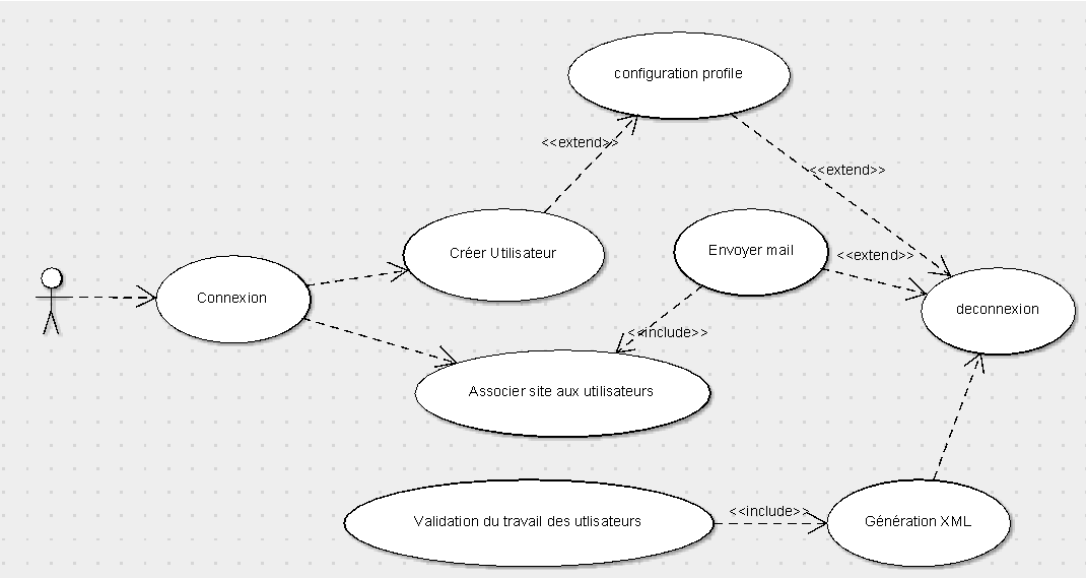



FIG. B.4 – Diagramme de cas d'utilisation - Administrateur

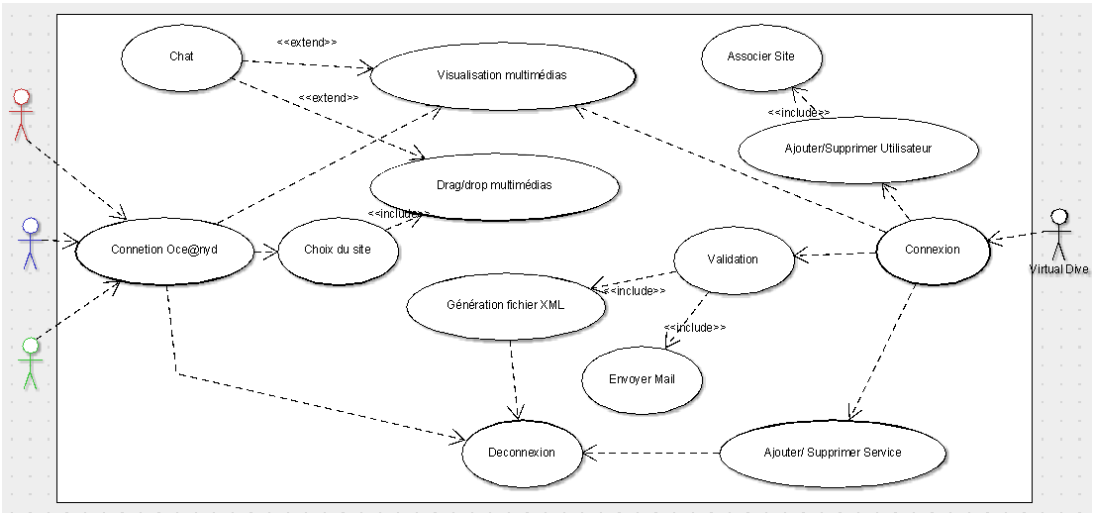



FIG. B.5 – Diagramme de cas d'utilisation pour tout les acteurs du système

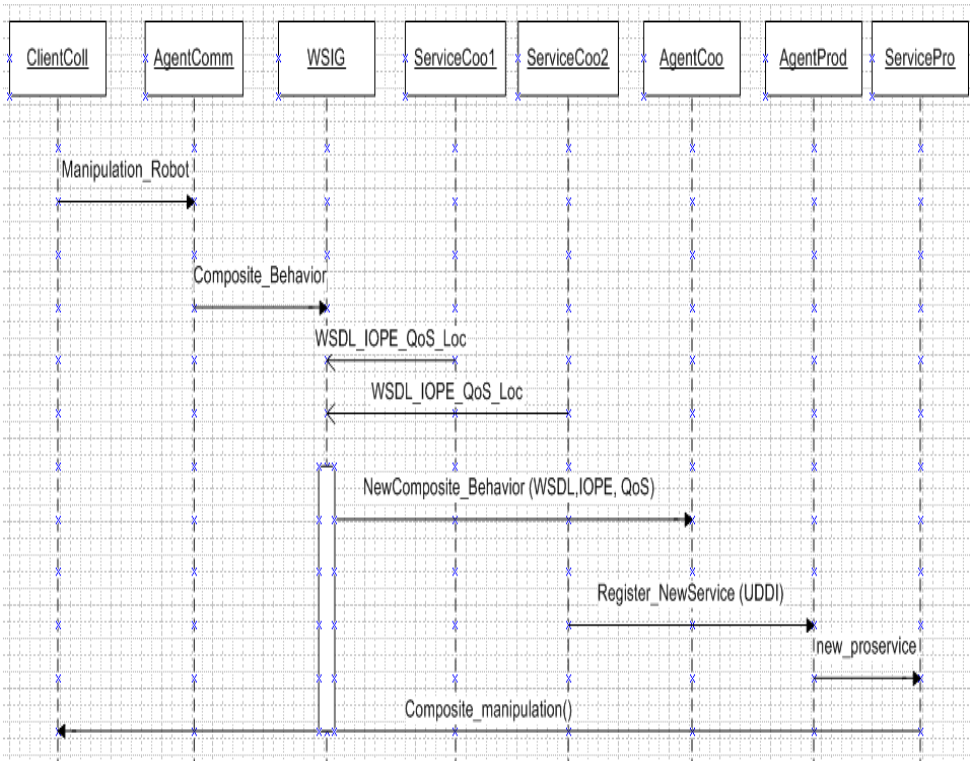



FIG. B.6 – Diagramme de séquence pour la création d'un service de manipulation

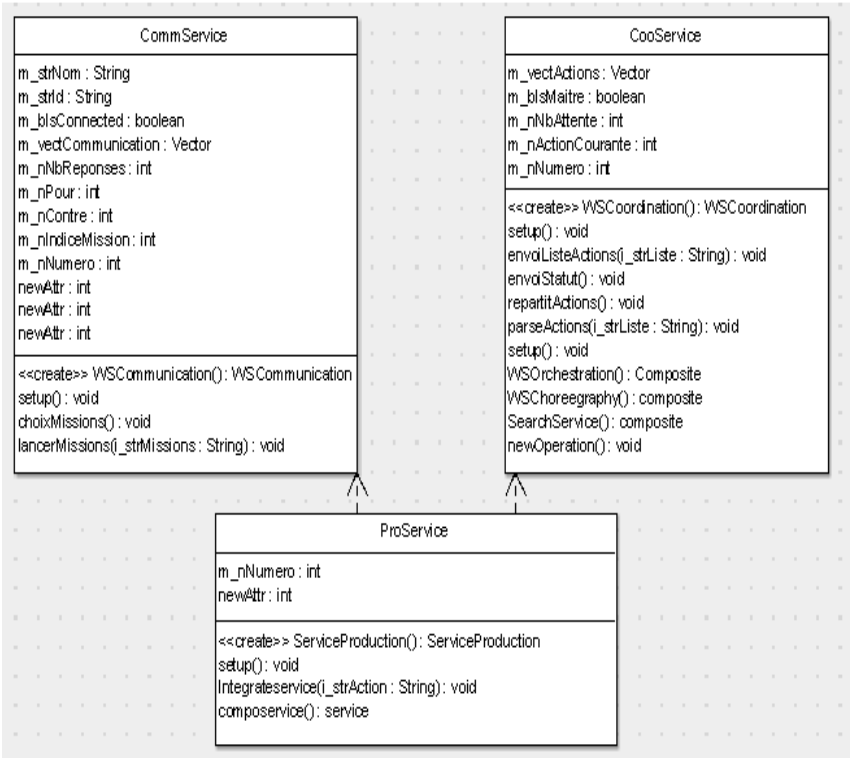



FIG. B.7 – Classes des services du système selon le modèle 3C

### B.1.2 Conception de la bases de données

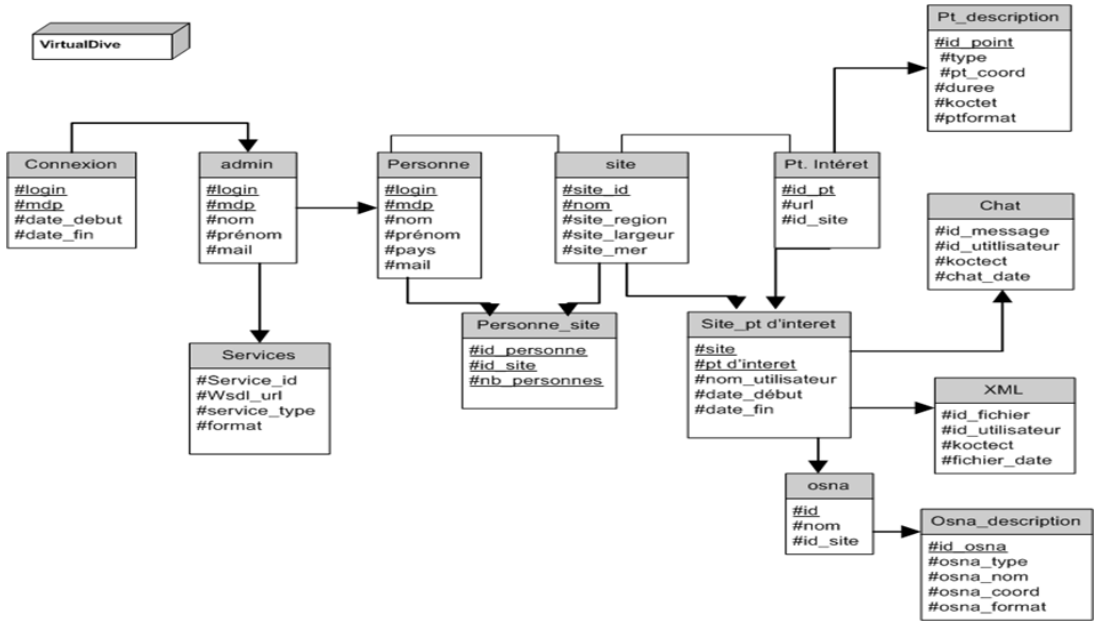



FIG. B.8 – Conception de la bases de données pour l'adminstrateur de l'application

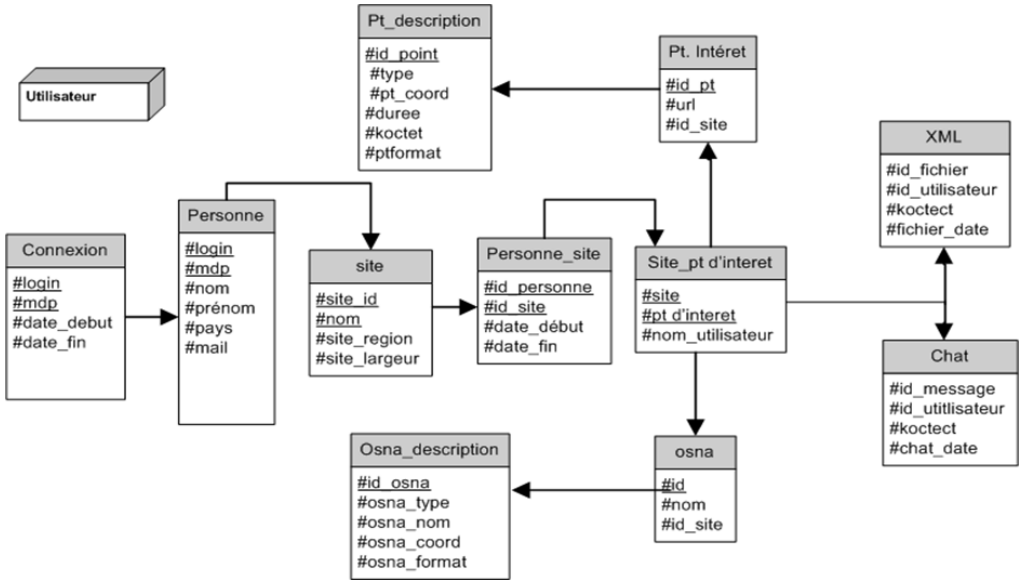



FIG. B.9 – Conception de la bases de données pour l'utilisateur de l'application

### B.1.3 Questionnaire et résultat des évaluations de *OceanydGroupware*

**Utilisabilité** :

1. L'upload dun fichier (image, vidéo, son) vous semble t-il facile ?
   (a) Oui
   (b) Non
   (c) Peut être

2. Le déplacement d'un fichier sur la carte vous semble t-il facile ?
   (a) Oui
   (b) Non
   (c) Peut être
3. J'ai réussi à déplacer le fichier dès les premiers instants ?
   (a) Oui
   (b) Non
   (c) Peut être
4. J'ai réussi à dessiner une trajectoire ?
   (a) Oui
   (b) Non
   (c) Peut être
5. J'ai réussi à créer une surface ?
   (a) Oui
   (b) Non
   (c) Peut être
6. Les explications d'utilisation sur l'interface vous semblent utiles ?
   (a) Oui
   (b) Non
   (c) Peut être

**Satisfaction** :

1. Pensez vous que les fichiers uploadés sont correctement affichés ?
   (a) Oui
   (b) Non
   (c) Peut être
2. Pensez vous que les informations affichées à l'écran sont trop nombreuses ?
   (a) Oui
   (b) Non
   (c) Peut être
3. Pensez vous que les informations affichées à l'écran sont pertinentes ?
   (a) Oui
   (b) Non
   (c) Peut être
4. Avez vous eu conscience de ce que voulais faire l'autre utilisateur en déplacent le fichier sur la carte ?
   (a) Oui
   (b) Non
   (c) Peut être
5. Avez vous eu conscience de ce que voulais faire l'autre utilisateur en déplacent le fichier sur la carte ?
   (a) Oui
   (b) Non
   (c) Peut être

**Collaboration** :

1. Je pense que l'autre participant observe mes actions virtuellement ?
   (a) Oui
   (b) Non
   (c) Peut être
   (d) Je ne sais pas
2. Les trajectoires effectuées par les autres participants étaient claires ?
   (a) Oui
   (b) Non
   (c) Peut être
   (d) Je ne sais pas
3. Mes actions étaient dépendantes des actions des autres participants ?
   (a) Oui
   (b) Non
   (c) Peut être
   (d) Je ne sais pas
4. Pensez vous que l'intégration d'un service (météo, informations sur la région sous-marine etc.) dans l'interface pourra apporter une meilleur utilisation de l'application ?
   (a) Oui
   (b) Probablement Oui
   (c) Non
   (d) Probablement Non
   (e) Je ne sais pas
5. Pensez vous que la composition de services (notamment, la création d'une surface à partir de composition de points) est utile pour la créer une information pertinente ?
   (a) Oui
   (b) Probablement Oui
   (c) Non
   (d) Probablement Non
   (e) Je ne sais pas
6. Pensez vous que la composition de services (notamment, la création d'une surface à partir de composition de points) est utile pour la créer une information pertinente ?
   (a) Oui
   (b) Probablement Oui
   (c) Non
   (d) Probablement Non
   (e) Je ne sais pas

**Appréciation** :

1. J'ai été un participant très présent dans les phases de dialogue (chat) ?
   (a) Oui
   (b) Non
   (c) Peut être
   (d) Je ne sais pas

2. Avez-vous apprécié travailler sur l'interface ?
   (a) Oui
   (b) Non
   (c) Peut être
   (d) Je ne sais pas
3. Avez-vous apprécié travailler sur l'interface ?
   (a) Oui
   (b) Non
   (c) Peut être
   (d) Je ne sais pas
4. Quelles améliorations pourrez-vous suggéré pour une meilleure utilisation de l'interface ?

#### B.1.3.1 Résultats de l'évaluation subjective de *OceanydGroupware*

| | Q1 | | | Q2 | | | Q3 | | | Q4 | | | Q5 | | | Q6 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | a | b | c | a | b | c | a | b | c | a | b | c | a | b | c |
| Personne 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Personne 2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 3 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Personne 4 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| Personne 6 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| Personne 7 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 8 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| Personne 9 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 10 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| Total | 6 | 2 | 2 | 10 | 0 | 0 | 10 | 0 | 0 | 9 | 1 | 0 | 7 | 2 | 1 | 4 | 2 | 4 |

FIG. B.10 – Résultats de l'évaluation de Oceanyd Groupware pour la partie Utilisabilité

| | Q1 | | | Q2 | | | Q3 | | | Q4 | | | Q5 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | a | b | c | a | b | c | a | b | c | a | b | c |
| Personne 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 2 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Personne 3 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 4 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Personne 5 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| Personne 6 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 7 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 8 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Personne 9 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Personne 10 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Total | 6 | 2 | 2 | 2 | 8 | 0 | 2 | 7 | 1 | 6 | 3 | 1 | 10 | 0 | 0 |

FIG. B.11 – Résultats de l'évaluation de Oceanyd Groupware pour la partie Satisfaction

| | Q1 | | | | Q2 | | | | Q3 | | | | Q4 | | | | | Q5 | | | | | Q6 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | a | b | c | d | a | b | c | d | a | b | c | d | e | a | b | c | d | e | a | b | c | d | e |
| Personne 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 3 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Personne 4 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Personne 5 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 6 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 7 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 8 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Personne 9 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 10 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total | 4 | 0 | 2 | 4 | 8 | 2 | 0 | 0 | 3 | 5 | 2 | 0 | 5 | 4 | 0 | 0 | 1 | 3 | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 7 | 2 |

FIG. B.12 – Résultats de l'évaluation de Oceanyd Groupware pour la partie Collaboration

| | Q1 | | | | Q2 | | | | Q3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | a | b | c | d | a | b | c | d |
| Personne 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Personne 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Personne 3 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Personne 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Personne 5 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Personne 6 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Personne 7 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Personne 8 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Personne 9 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Personne 10 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Total | 5 | 5 | 0 | 0 | 9 | 0 | 1 | 0 | 7 | 2 | 1 | 0 |

FIG. B.13 – Résultats de l'évaluation de Oceanyd Groupware pour la partie Appréciation

### B.1.4 Ontologie et sémantique d'information

Notre architecture offre plusieurs fonctionnalités génériques, qui peuvent être exploitées dans les deux domaines d'application présentés dans ce chapitre : La multimédia et la téléopération sur internet. Cependant, chaque domaine d'application possède ses fonctionnalités privées, comme les commandes de manipulation de robot pour la téléopération, ainsi que les téléchargements des fichiers multimédias (images, vidéo et audio) pour le projet DIGITAL OCEAN. Ainsi, la malléabilité intervient dans le noyau de l'application générique. L'idée donc c'est d'avoir une intégration dynamique de nouvelles fonctionnalités génériques au niveau du serveur, qui peuvent être exploité indépendamment du domaine de l'application. En effet, l'idée c'est de mettre en œuvre certains services génériques (comme le chat), et les exposer comment étant des services web au monde externe. Ainsi, les utilisateurs pourront réutiliser ses services dans leurs propres applications. Dans le domaine du génie logiciel, une ontologie représente une conceptualisation partagée et convenue d'un domaine. L'utilisation des ontologies a gagné de la popularité ces dernières années vue qu'elles facilitent l'interopérabilité et le raisonnement de la machine. Le terme d'ontologie implique également la modélisation des règles du domaine, qui va fournir un niveau supplémentaire de "compréhension" pour la machine.


FIG. B.14 – Ontologie de l'architecture U3D

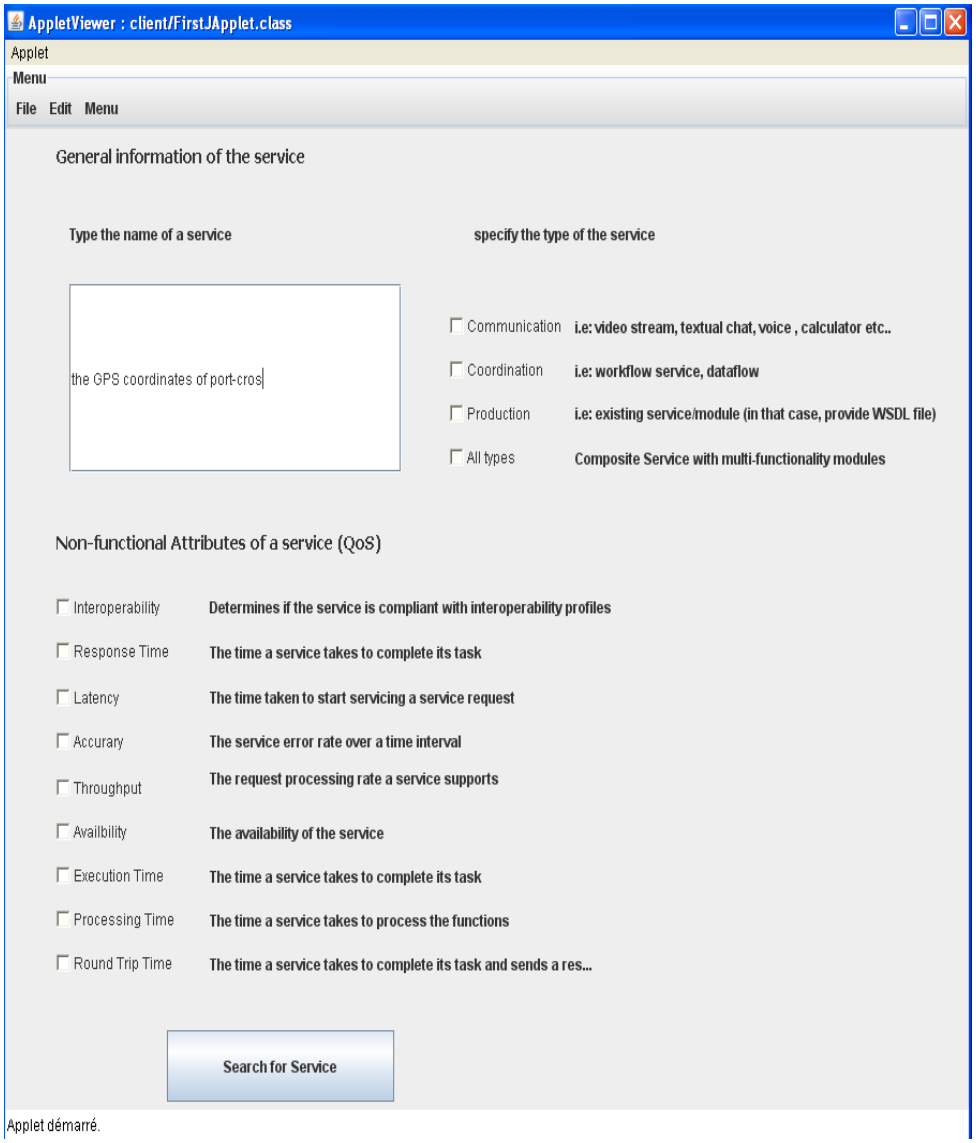



FIG. B.15 – Interface pour recherche un service à partir de ses attributs non-fonctionnels (QoS).