



Master Informatique deuxième année

-

Rapport de Stage

Johan Herrmann

Encadrant : Christophe Lang

2010

Remerciements

Avant de commencer ce rapport, je tiens à remercier toutes les personnes qui ont contribué au bon déroulement de mon stage :

- Christophe Lang, maître de conférence au LIFC, que je tiens à remercier particulièrement pour son soutien, son aide et ses conseils tout au long de ce stage,
- Pierre-Marie Gruss, ingénieur chez Em@systec, pour sa disponibilité et sa patience,
- Fabrice BOUQUET, professeur au LIFC, pour avoir accepté de participer au jury de présentation de ces travaux.

Je tiens également à remercier l'ensemble de mes collègues avec qui j'ai été en stage pour la bonne ambiance qui régnait au sein des stagiaires.

Je finirai en remerciant mes proches qui m'ont soutenu tout au long de ce stage :

- mes parents et grands parents pour leurs encouragements et leur aide,
- mes amis, tout particulièrement, Flavie, Guillaume, Benjamin et Sarah pour leur soutien et leur écoute dans les moments difficiles et surtout pour leur compagnie lors des bons moments.

Table des matières

| | | |
|------------|--|-----------|
| I | Introduction | 8 |
| II | État de l’art | 10 |
| 1 | La e-maintenance et la s-maintenance | 10 |
| 1.1 | La maintenance distribuée | 10 |
| 1.1.1 | Les systèmes de maintenance | 10 |
| 1.1.2 | Présentation de la e-maintenance | 11 |
| 1.1.3 | Les atouts de la e-maintenance | 12 |
| 1.1.4 | Conclusion | 13 |
| 1.2 | La maintenance sémantique | 13 |
| 1.2.1 | La capitalisation des connaissances | 14 |
| 1.2.2 | La sémantique dans la maintenance | 14 |
| 1.3 | Conclusion | 15 |
| 2 | Les ontologies et Le web sémantique | 16 |
| 2.1 | Ressource Description Framework | 16 |
| 2.2 | Ontology Web Language | 17 |
| 2.3 | OWL dans SMAC | 17 |
| 3 | La collaboration dans les systèmes de e-maintenance | 17 |
| 3.1 | Définition de la coopération et de la collaboration | 18 |
| 3.1.1 | La coopération | 18 |
| 3.1.2 | La collaboration | 19 |
| 3.2 | Les systèmes de maintenance coopératifs et collaboratifs | 19 |
| 3.2.1 | La méthode “Scoop” pour les systèmes coopératifs | 19 |
| 3.2.2 | Les systèmes collaboratifs | 22 |
| 4 | Synthèse | 24 |
| III | Présentation du projet SMAC | 26 |
| 5 | Le projet SMAC | 26 |
| 5.1 | L’interopérabilité sémantique | 26 |
| 5.2 | L’ontologie et la capitalisation des connaissances | 27 |
| 5.3 | La réalité augmentée | 27 |
| 5.4 | La collaboration | 27 |

| | | |
|---|--|-----------|
| 5.5 | Conclusion | 28 |
| 6 | Système cible | 28 |
| 7 | Synthèse | 30 |
| IV Vers une modélisation agent de la s-maintenance | | |
| 32 | | |
| 8 | Choix du paradigme Agent | 32 |
| 9 | Sémantique dans les échanges inter-agent | 33 |
| 9.1 | L'ontologie SMAC | 33 |
| 9.1.1 | Le produit | 35 |
| 9.1.2 | Le cycle de vie | 35 |
| 9.1.3 | Les actions | 35 |
| 9.1.4 | Conclusion | 36 |
| 9.2 | Les spécifications de la FIPA | 36 |
| 9.2.1 | La spécification FIPA-ACL | 36 |
| 9.2.2 | Les paramètres des message | 37 |
| 9.2.3 | La spécification FIPA-OSS | 38 |
| 9.3 | Conclusion | 38 |
| 10 | AML et systèmes complexes | 39 |
| 10.1 | Agent Modeling Language - AML | 39 |
| 10.1.1 | Concepts Structurels | 39 |
| 10.1.2 | Concepts sociaux | 40 |
| 10.2 | Modélisation agent des systèmes complexes. | 42 |
| 10.2.1 | Les restrictions d'AML | 43 |
| 10.2.2 | Le modèle de structure | 44 |
| 10.3 | Conclusion | 45 |
| 11 | Modèle SMA pour la simulation de la collaboration | 45 |
| 11.1 | Structure organisationnelle | 46 |
| 11.1.1 | Niveau 0 | 46 |
| 11.1.2 | Niveau 1 | 46 |
| 11.1.3 | Niveau 2 | 48 |
| 11.2 | Structure sociale | 48 |
| 11.2.1 | Niveau 0 | 48 |
| 11.2.2 | Niveau 1 | 48 |
| 11.2.3 | Niveau 2 | 48 |

| | | |
|-----------|--|-----------|
| 11.3 | Relation entre l'ontologie et la structure du modèle | 51 |
| 11.4 | Conclusion | 51 |
| 12 | Synthèse | 52 |
| V | Implémentation | 54 |
| 13 | Choix de la plate-forme multi-agents | 54 |
| 13.1 | MadKit | 54 |
| 13.2 | JADE | 55 |
| 13.2.1 | Les greffons de JADE | 57 |
| 13.3 | Choix | 58 |
| 14 | Implémentation de l'AGR dans JADE | 58 |
| 14.1 | GroupAGRManager | 59 |
| 14.2 | GroupAGR et RoleAGR | 59 |
| 14.3 | AgentAGR | 60 |
| 14.4 | Le package SW | 60 |
| 14.5 | Conclusion | 61 |
| 15 | Implémentation de notre modèle et difficultés rencontrées | 61 |
| 15.1 | Les problèmes avec l'ontologie et l'Ontology Bean Generator . | 61 |
| 15.2 | Les éléments partiellement implémentés | 62 |
| 15.2.1 | L'OntologyAgent | 62 |
| 15.2.2 | L'agent Humain | 62 |
| 15.2.3 | La classe capteur | 62 |
| 15.2.4 | Les classes SCADA et MonitorSCADA | 62 |
| 15.3 | Conclusion | 63 |
| 16 | Synthèse | 63 |
| VI | Conclusion | 64 |

Table des figures

| | | |
|----|--|----|
| 1 | Intégration de la e-maintenance [Marquez, 2007] | 12 |
| 2 | Classification de différentes architectures en maintenance [Rasovska, 2006] | 15 |
| 3 | Différence entre la coopération et la collaboration [Seguy, 2008] | 18 |
| 4 | Intéactions entre les différents composants d'un système de e-maintenance [Saint-Voirin, 2006] | 20 |
| 5 | Relation entre les agents d'un système générique de e-maintenance [Trappey et al., 2008] | 23 |
| 6 | Exemple de réalité augmentée | 28 |
| 7 | Le système cible | 29 |
| 8 | L'ontologie de SMAC [Aristeidis, 2010] | 34 |
| 9 | Type d'agent AML | 40 |
| 10 | Type de ressource AML | 40 |
| 11 | Type d'unité organisationnelle AML | 41 |
| 12 | Type de rôle AML | 41 |
| 13 | Propriétés Sociales AML | 41 |
| 14 | Fragment de comportement AML | 42 |
| 15 | Méta-modèle de structure des systèmes complexes [Brocard, 2010] | 43 |
| 16 | Structure organisationnelle Niveau 0 | 47 |
| 17 | Structure organisationnelle Niveau 1 | 49 |
| 18 | Structure organisationnelle Niveau 2 | 50 |
| 19 | Structure sociale niveau 0 | 50 |
| 20 | Structure sociale niveau 1 | 50 |
| 21 | Structure sociale niveau 2 | 51 |
| 22 | Architecture générale de MadKit [Gutknecht et al., 2000] | 55 |
| 23 | Architecture de JADE [Bellifemine et al., 2001] | 56 |
| 24 | Architecture interne de JADE [Bellifemine et al., 2002] | 56 |

Liste des tableaux

| | | |
|---|--|----|
| 1 | FIPA ACL Message Parameters [FIPA, 2002] | 37 |
| 2 | Correspondances des modèles | 51 |

Première partie

Introduction

Nous avons effectué un stage à l'issue de notre formation en Master Informatique. Pendant ce stage, nous avons fait partie de l'équipe de recherche du projet SMAC. Ce projet a pour ambition le développement d'une plate-forme de maintenance sémantique ou s-maintenance.

La maintenance est un "ensemble d'opérations permettant de maintenir ou de rétablir un matériel, un appareil, un véhicule, etc., dans un état donné, ou de lui restituer des caractéristiques de fonctionnement spécifiées" [Larousse, 2010]. La s-maintenance est basée sur la e-maintenance elle même basée sur la télémaintenance.

Le terme télémaintenance signifie littéralement maintenance à distance ; c'est le fait de pouvoir effectuer des opérations de maintenance sans être physiquement présent sur le lieu de la maintenance. La e-maintenance reprend cette notion de distance mais y ajoute la capacité de forte distribution du réseau Internet. Elle s'articule autour de l'utilisation de web services pour l'interconnexion et l'interopérabilité entre les différentes parties de la plate-forme. Les plates-formes de e-maintenance sont conçues pour faciliter la coopération des acteurs externalisés et géographiquement répartis.

La s-maintenance propose d'améliorer encore le concept de e-maintenance en se concentrant sur ces trois points essentiels que sont l'interopérabilité sémantique, la capitalisation des connaissances et la collaboration au sein de la plate-forme. L'interopérabilité sémantique permet de faire dialoguer des parties distantes de la plate-forme à un niveau conceptuel. La capitalisation des connaissances permet d'affiner les diagnostics et de pouvoir prévoir avec toujours plus de finesse les détériorations futures que subiront les produits maintenus. La collaboration entre les acteurs de la maintenance découle de l'interopérabilité sémantique qui facilite le travail en commun au sein de la plate-forme.

Le but de notre stage était d'éprouver les notions d'interopérabilité sémantique de collaboration en créant un simulateur de cette plate-forme et en y incorporant une sémantique forte ainsi que des algorithmes de collaboration.

Dans un premier temps, nous nous sommes concentrés à nous familiariser avec les concepts présents dans le projet et à rechercher quels avaient

été les travaux précédemment menés dans ce domaine. La partie II “État de l’art” de ce rapport est le fruit de ce travail. Nous avons découpé cette partie en trois. Dans une première section nous présentons la e-maintenance et la s-maintenance. Dans la deuxième section nous détaillons le concept d’ontologie qui est le porteur de l’interopérabilité sémantique recherché dans SMAC. Enfin, nous dressons un panorama des systèmes de maintenance coopérative et collaborative existant.

Une fois immergés dans l’univers de la maintenance, nous présenterons plus en détail le projet SMAC avec ses aspects théoriques et pratiques dans la partie III.

Toutes les clés permettant de la comprendre ayant été données, nous détaillerons notre contribution théorique dans la partie IV. Dans cette partie, nous expliquerons notre choix du paradigme multi-agents pour la modélisation et l’implémentation du simulateur. Nous nous attarderons ensuite sur la sémantique dans les échanges inter-agents afin de modéliser fidèlement l’interopérabilité sémantique. Nous considérons la plate-forme SMAC comme un système complexe. Nous allons donc présenter une méthode de représentation de système complexe à base d’agents. Et, dans la dernière section de cette partie nous présenterons la plate-forme SMAC modélisée avec cette méthode.

Enfin, La partie V présente le travail d’implémentation effectué pour la création du simulateur. Une première section détaillera le choix de la plate-forme multi-agents retenue. Dans la deuxième nous présenterons le développement du modèle Agent Groupe Rôle dans le système multi-agents JADE. Pour finir nous ferons un point sur l’état d’avancement de l’implémentation et des causes de son arrêt.

Pour terminer nous synthétiserons l’ensemble du travail effectué dans la conclusion de ce rapport.

Deuxième partie

État de l'art

Cette partie présente le travail de recherche mené en vue d'acquérir la connaissance nécessaire à la compréhension des aspects importants du projet SMAC qui seront détaillés dans la partie III.

Dans un premier temps, nous présenterons la e-maintenance dans un cadre général, puis nous nous intéresserons plus spécifiquement aux travaux menés dans la maintenance sémantique. Nous élargirons ensuite nos recherches aux techniques du web sémantique qui seront l'un des constituants importants du projet SMAC pour l'élaboration d'une ontologie regroupant les connaissances du système. Nous continuerons par analyser les procédés de collaboration ainsi que les travaux la liant à la maintenance. Nous terminerons par une synthèse du travail effectué.

1 La e-maintenance et la s-maintenance

La e-maintenance est actuellement le système de maintenance le plus évolué du marché. Cette section a pour but dans un premier temps de présenter ce qu'est la e-maintenance et ensuite de s'intéresser aux aspects sémantiques des informations des plates-formes de e-maintenance pour leur intégration à nos travaux.

1.1 La maintenance distribuée

Nous allons aborder dans cette partie les généralités sur la e-maintenance. Nous commencerons par exposer les différents types de maintenance. Nous dresserons ensuite une description générale de la e-maintenance. Pour finir nous présenterons ses atouts puis nous conclurons.

1.1.1 Les systèmes de maintenance

Nous pouvons distinguer quatre types de maintenance d'après [Afnor, 2001] :

- Corrective : dépannage, réparation.

La maintenance corrective est le type de maintenance le mieux connu car historiquement, la maintenance était uniquement apparentée à la

réparation. Il s'agit ici de traitement de maintenance curative pour remettre en état un équipement qui a cessé de fonctionner de manière inattendue.

- Préventive : anticipation des pannes.
Ce type de maintenance est concentré sur la prévision des pannes, on change les pièces préventivement, conformément à leur durée de vie.
- Pro-active : anticipation des pannes en tenant compte de l'historique.
La maintenance pro-active ressemble à la maintenance préventive mais ajoute un élément non négligeable qui est une conservation de l'historique des pannes. Cet historique est pris en compte dans le processus de prise de décision pour une intervention sur un équipement.
- Améliorative : amélioration des fonctionnalités ou de la fiabilité d'un équipement (ex : ajout d'un capteur).
C'est la forme de maintenance la plus rarement utilisée car elle fait partie des interventions de grande maintenance, comme par exemple une remise à neuf.

Nous pouvons également en ajouter une autre, la self-maintenance qui donne la possibilité aux équipements de se réparer eux-mêmes [Lee J, 2008]. Ce type de maintenance n'est évidemment pas possible pour tous les équipements ou pour la totalité d'un équipement. Il sera plus facile par exemple qu'une machine effectue des opérations sur son logiciel que sur son matériel n'ayant pas la capacité physique de changer ses propres pièces.

Les systèmes de maintenance sont des systèmes informatiques qui ont pour but d'aider l'entreprise à gérer les processus de maintenance de la manière la plus efficace possible [Rasovska, 2006]. Les stratégies adoptées par ces systèmes intègrent tout ou partie des types de maintenance. La e-maintenance représente à l'heure actuelle un des systèmes les plus évolués de maintenance.

1.1.2 Présentation de la e-maintenance

Selon [Marquez, 2007] "La e-maintenance est un concept qui peut être défini comme un support de maintenance qui inclut les ressources, les services et la gestion nécessaires pour rendre possible l'exécution des processus de décision pro-active". Ce support s'appuie sur les technologies web, es-

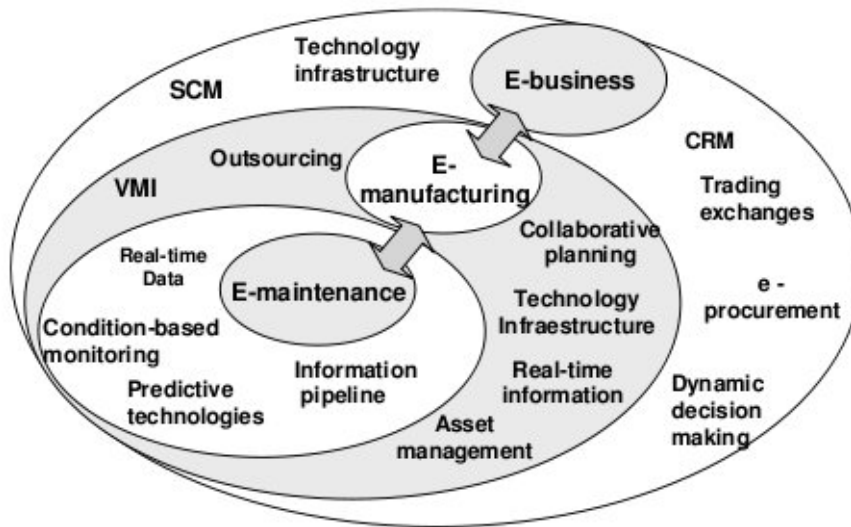


FIGURE 1 – Intégration de la e-maintenance [Marquez, 2007]

sentiellement sur les web services, et utilisant ou non l'internet. Contrairement à la vision traditionnelle où la maintenance est considérée comme un coup additionnel, les nouvelles techniques de maintenance sont intégrées dans le management global de l'entreprise pour l'amélioration des coups de production et de la fiabilité des équipements. La e-maintenance est donc à son niveau intégrée avec les processus de e-production et de e-business (cf. fig. 1).

1.1.3 Les atouts de la e-maintenance

La e-maintenance permet diverses actions en lien avec son architecture distribuée et axée sur les services.

- La maintenance à distance
Au moyen d'équipements nomades (téléphone portable, pda), un technicien ou un expert peut se connecter à une plate-forme de e-maintenance et effectuer diverses opérations comme des contrôles, des réglages ou des diagnostics.
- La maintenance coopérative
De part sa nature distribuée et distante, la e-maintenance a été conçue pour la coopération entre les différents acteurs de la maintenance que sont les clients, les experts, les techniciens, etc. En d'autres mots, elle

facilite le flux bidirectionnel de données et d'informations dans les processus de prise de décision et de prévision, et ce à tous les niveaux [Ucar and Qiu, 2005].

- La maintenance immédiate

Un des objectifs de la e-maintenance est de fournir les informations venant des meilleures sources possibles au meilleur moment et aux meilleurs acteurs. La collaboration des acteurs au moment adéquat permet une grande réactivité lors de la survenue d'opérations (programmées ou non) de maintenance. Pour se faire, les plates-formes de e-maintenance disposent de systèmes de surveillance temps réel de l'état des équipements et des systèmes d'alerte ainsi que des relations avec des agents (humains ou informatiques) experts. La résultante en est un haut taux de communication qui permet d'obtenir plusieurs expertises rapidement [Garcia et al., 2004].

- La maintenance prédictive

Une des principales problématiques de la e-maintenance est de fournir des outils de maintenance prédictive [Lee, 2003]. Grâce aux équipements de surveillance et de "l'intelligence" de la plate-forme, les entreprises peuvent ainsi obtenir des pronostics sur l'état de dégradation des équipements [Koc et al., 2003].

1.1.4 Conclusion

Nous avons vu les différents types de maintenance et remarqué que les systèmes de e-maintenance permettaient de les gérer de manière souple et distribuée avec l'utilisation de web-services. Nous avons précisé que la e-maintenance se devait de fournir de l'aide à la décision de manière pro-active ainsi que de permettre la coopération entre les acteurs. Pour le projet SMAC ces deux aspects de la maintenance sont améliorables grâce à l'utilisation d'une sémantique riche dans les communications au sein de la plate-forme.

1.2 La maintenance sémantique

Les systèmes de e-maintenance fournissent de multiples aides aux diagnostics lors des phases de maintenance. Pour améliorer ces diagnostics, le système a besoin de pouvoir acquérir de l'expérience sur ses actions et ses diagnostics par un processus d'apprentissage. Cela implique que le système doit "comprendre" les informations qu'il traite. Ces informations ont besoin

d'une sémantique que la machine puisse comprendre pour les traiter de manière intelligente et les partager avec les acteurs/agents de la maintenance. Nous allons donc présenter la notion de capitalisation de connaissances dans une première section et quelques aspects des travaux de [Rasovska, 2006] dans une seconde partie dédiée à la sémantique dans la maintenance.

1.2.1 La capitalisation des connaissances

La capitalisation des connaissances consiste à réutiliser, de façon pertinente, la connaissance d'un domaine donné, précédemment stockée et modélisée, afin d'effectuer de nouvelles tâches [Simon, 1996]. La capitalisation des connaissances dans un système de e-maintenance (et à plus forte raison de s-maintenance) a donc pour but de réutiliser le savoir du système pour affiner et accélérer les diagnostics. La littérature offre un grand nombre de modèles de gestion des connaissances qui sont tous dérivés d'un modèle générique de [Lai and Chu, 2000] bâti autour de huit phases :

- Initiation
- Élaboration
- Modélisation
- Préservation
- Distribution et stockage
- Utilisation
- Révision

La modélisation concerne l'organisation et la représentation des connaissances. La distribution et le transfert consistent à savoir comment partager la connaissance aux utilisateurs finaux. L'utilisation consiste à savoir comment utiliser la connaissance pour produire de la valeur ajoutée.

Pour qu'une machine sache se charger de ces tâches il est nécessaire qu'elle comprenne les informations qu'elle traite. Ces informations doivent avoir un sens, une sémantique.

1.2.2 La sémantique dans la maintenance

A l'exception des travaux de [Rasovska, 2006], nous n'avons pas trouvé de programme ou d'article de recherche traitant de maintenance sémantique. De nombreux travaux ont été réalisés sur le web sémantique. La s-maintenance s'appuie sur le web sémantique pour l'ajout de sens aux informations traitées via une ontologie.

Selon [Rasovska et al., 2007], "l'architecture d'une plate-forme de s-maintenance prend appui sur l'architecture d'e-maintenance où l'interopérabi-

lité des différents systèmes intégrés dans la plate-forme est garantie par un échange de connaissances représentées par une ontologie”. Comme on peut le voir sur la figure 2 la s-maintenance introduit une réelle collaboration en unissant les compétences et connaissances des différents acteurs afin de répondre aux besoins en maintenance.

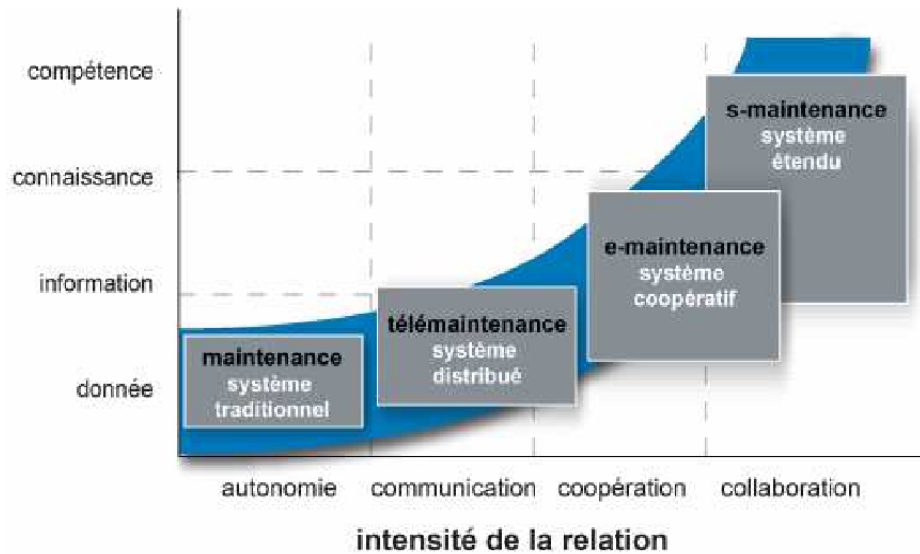


FIGURE 2 – Classification de différentes architectures en maintenance [Rasovska, 2006]

1.3 Conclusion

Cette section nous a permis de présenter la maintenance et plus particulièrement la e-maintenance et la s-maintenance.

Nous avons détaillé en quoi consistait la maintenance au travers des différents types de maintenance rencontrés dans la littérature ce qui nous a permis d’appréhender les concepts de e-maintenance. Nous avons vu également les limitations de la e-maintenance quant à la compréhension par le système de ses actions. Ces limitations ont donné lieu à un nouveau concept : celui de la s-maintenance qui introduit la sémantique dans le système au moyen d’une ontologie.

2 Les ontologies et Le web sémantique

Le terme “ontologie” a beaucoup de sens différents et beaucoup de nuances de sens suivant le domaine dans lequel il est utilisé. L’ontologie, en philosophie, est une branche de la métaphysique qui concerne l’étude de l’être en tant qu’être, de ce qui le définit. En informatique, par extension à la définition philosophique, une ontologie est un objet logiciel complexe définissant les concepts existant au sein d’un système et les relations qui les lient. Pour un agent logiciel, une ontologie peut être vue comme la représentation de son univers .

Le développement des ontologies en informatique s’est notamment fait avec l’extension du web sémantique. D’après le W3C, le web sémantique est le web des données. Le web sémantique concerne deux sujets. Premièrement, il s’agit de formats communs pour l’intégration et la combinaison des données provenant de sources diverses. Deuxièmement, il s’agit d’un langage pour spécifier comment les données se rapportent à des objets du monde réel.

Cela permet à un humain ou une machine de commencer son exploration du web depuis une base de données puis de se déplacer dans d’autres bases de données qui ne sont plus reliées par leur connectique, mais à un niveau supérieur, par le fait qu’elles contiennent des objets de la même essence.

Cela signifie plus simplement que grâce au web sémantique les web services deviennent capables de manipuler non plus des données mais des concepts. Le cheminement vers une solution se fait alors plus “humainement” en passant d’un concept à un autre.

Le W3C propose plusieurs langages et recommandations pour le développement du web sémantique. Néanmoins nous n’en retiendrons que deux :

- Ressource Description Framework (RDF)
- Web Ontology Language (OWL)

2.1 Ressource Description Framework

RDF est une structure de données constituée de nœuds et organisée en graphe. Il est destiné à décrire de façon formelle les ressources Web et leurs méta-données. RDF est structuré sous la forme de triplet {sujet, prédicat, objet}. Le sujet est la ressource à décrire, le prédicat est un type de propriété applicable à cette ressource et l’objet est la valeur de cette propriété. Un objet peut être soit une donnée, soit une autre ressource, ce qui permet de faire des relations entre les ressources décrites.

RDF n’est pas associé à une syntaxe particulière, il est néanmoins courant

de voir des documents RDF écrit en XML, ce langage étant très répandu pour la structuration de données sur le web.

RDF peut être complété par le langage RDFS (S pour Shema). RDFS introduit la notion de classe, ainsi une ressource peut être considérée comme une classe. Avec le concept de classe il introduit également la hiérarchisation des classes et l'héritage.

RDF et RDFS sont les deux structures à l'origine de OWL.

2.2 Ontology Web Language

Le langage OWL est une extension de RDF et RDFS. “Aux concepts de classe, de ressource, de littéral et de propriétés des sous-classes, de sous-propriétés, de champs de valeurs et de domaines d'application déjà présents dans RDFS, OWL ajoute les concepts de classes équivalentes, de propriétés équivalentes, d'égalité de deux ressources, de leurs différences, du contraire, de symétrie et de cardinalité...” [Wikipédia, 2010].

2.3 OWL dans SMAC

OWL est le langage qui a été “naturellement” utilisé pour développer l'ontologie de SMAC. En effet, OWL est le langage ontologique le plus répandu sur le web. Il est de plus bien instrumentalisé au moyen du logiciel “Protégé” avec lequel il est aisé de développer une ontologie sans être obligé de “coder” en XML. (Pour plus de détails sur cette ontologie il faut se reporter à la section 9.1.)

Grâce à l'ontologie développée dans SMAC, la collaboration au sein du système devient possible.

3 La collaboration dans les systèmes de e-maintenance

La collaboration est, avec la sémantique, un des points clé du projet. Pour en bien comprendre l'importance nous commencerons par définir la coopération qui est le mode de fonctionnement privilégié de la e-maintenance (cf fig.2 page 15), puis la collaboration qui concerne dans une plus grande mesure la s-maintenance. Nous finirons par une présentation des travaux réalisés dans le domaine de la maintenance coopérative et collaborative essentiellement basée sur les systèmes multi-agents.

3.1 Définition de la coopération et de la collaboration

Nous allons voir dans cette section les définitions et les différences entre la coopération et la collaboration illustrées sur la figure 3.

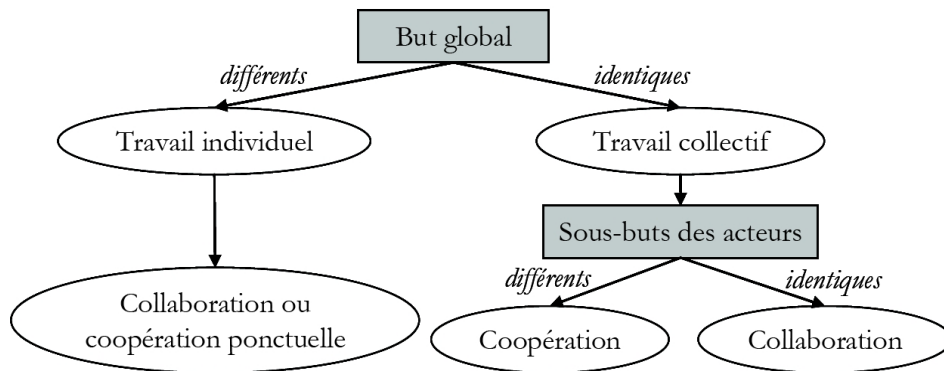


FIGURE 3 – Différence entre la coopération et la collaboration [Seguy, 2008]

3.1.1 La coopération

Nous allons commencer par définir l'action de coopérer puis voir son application dans la maintenance.

Coopérer c'est prendre part, concourir à une œuvre commune [Larousse, 2010]. Selon [Seguy, 2008], "un ensemble d'acteurs est dit coopérant si ces acteurs participent à une activité commune en réalisant individuellement une partie de cette activité. Il s'agit simplement d'une participation ponctuelle à une activité collective, sans engagement durable des acteurs qui coopèrent. L'activité collective est segmentée en activités individuelles, réparties entre les acteurs coopérants. Chaque acteur de la coopération garde son organisation, ses ressources et ses objectifs propres."

Les plates-formes de e-maintenance permettent de coopérer, de distribuer et partager de l'information en utilisant un réseau web [Rasovska, 2006]. Ces plates-formes servent à mutualiser les connaissances et à les regrouper dans un système d'information unique [Muller, 2005].

3.1.2 La collaboration

Comme pour la coopération, nous commencerons par sa définition puis nous verrons son application dans la maintenance.

Collaborer consiste à travailler de concert avec quelqu'un d'autre, l'aider dans ses fonctions ; participer avec un ou plusieurs autres à une œuvre commune [Larousse, 2010]. Toujours selon Seguy, “un ensemble d'acteurs est dit collaborant si ces acteurs travaillent ensemble à une production commune, avec un objectif commun et si chacun réalise une partie de l'activité du groupe.”

Traditionnellement, les plates-formes de e-maintenance permettent une forme de collaboration entre les personnes, les acteurs, avec l'objectif d'une décision commune. Le travail collaboratif est lié ici à la possibilité, pour un acteur local, de bénéficier de l'expertise d'autres acteurs externalisés. Néanmoins, la notion de collaboration est ici plus proche de celle de coopération. De plus, il ne s'agit ici que de la collaboration des acteurs utilisant le système et non pas des composants du système lui même.

3.2 Les systèmes de maintenance coopératifs et collaboratifs

Il existe dans la littérature plusieurs travaux sur les systèmes de maintenance coopératifs ou collaboratifs. Dans une première section nous présenterons la méthode “Scoop” développée en vue de modéliser les systèmes coopératifs et qui a une application à la e-maintenance. Dans la seconde nous présenterons divers travaux effectués sur la e-maintenance et la collaboration.

3.2.1 La méthode “Scoop” pour les systèmes coopératifs

[Saint-Voirin, 2006] a étudié la modélisation de systèmes coopératifs et son application à la e-maintenance (FIG 4). Il propose une méthodologie ainsi que des outils pour la modélisation, la mise en œuvre et la simulation de systèmes coopérants génériques : “Scoop”(Simulation des Systèmes coopératifs).

En confrontant les définitions données dans les points 3.1.1 et 3.1.2, la collaboration peut être vue comme une coopération renforcée d'une intention d'action collective. Nous ne nous sommes pas servis de ces travaux tels quels mais nous nous en sommes inspiré (essentiellement en ce qui concerne

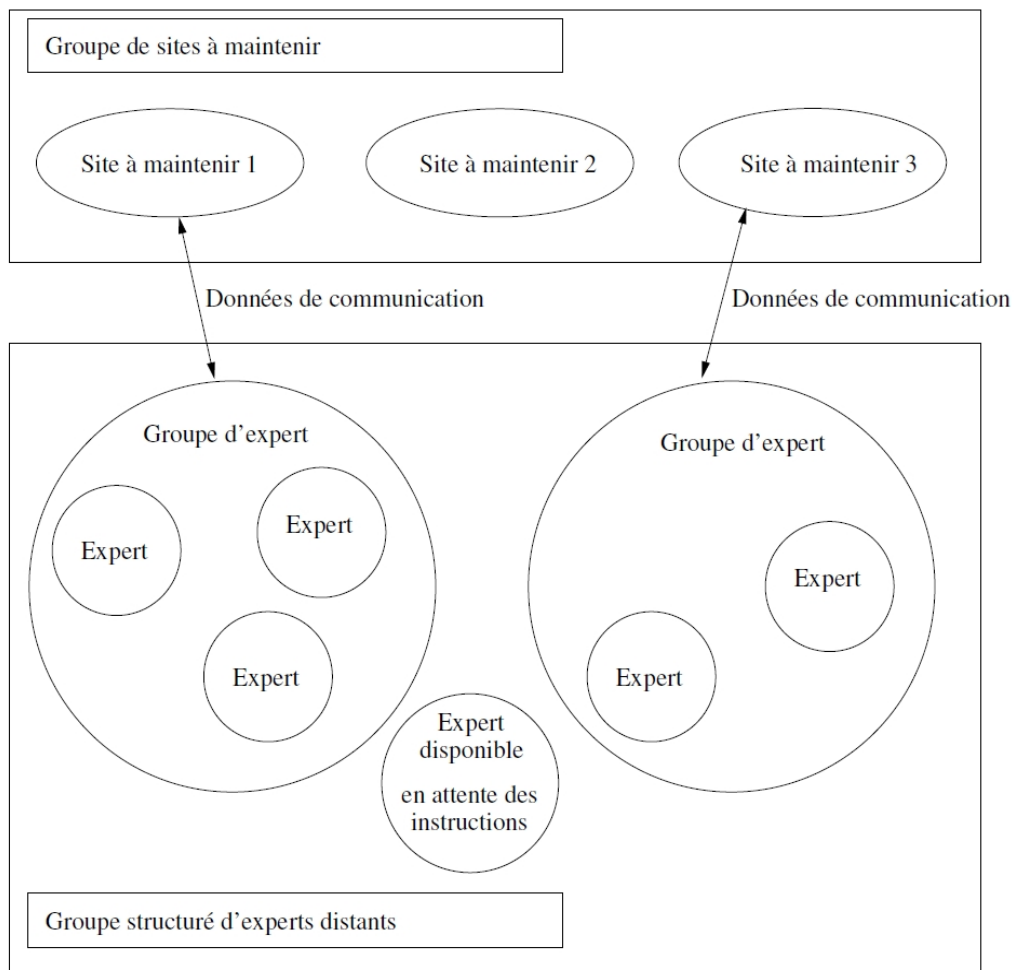


FIGURE 4 – Interactions entre les différents composants d'un système de e-maintenance [Saint-Voirin, 2006]

la partie multi-agents) pour modéliser notre simulateur ; nous pensons donc qu'il est intéressant de les présenter dans cette section.

La méthodologie Scoop

Cette méthodologie se décompose en cinq phases :

- besoins préalables
- besoins finaux
- conception architecturale
- conception détaillée

- implémentation

La phase “besoins préalables” consiste en une définition et un découpage des systèmes coopératifs.

La phase “besoins finaux” est un découpage plus fin des définitions précédentes

La phase “conception architecturale” utilise les deux phases précédentes pour définir un méta modèle de structure du système. C’est dans cette phase qu’est définie la nomenclature complète des acteurs du système et de leurs interactions.

La phase “conception détaillée” comporte plusieurs niveaux :

- la modélisation des interactions entre les acteurs par des réseaux de Petri colorés.
- la modélisation et la définition des connaissances/croyances des acteurs par des fichiers XML basés sur une DTD spécifique au système. Ce formalisme a été utilisé par Saint-Voirin car “les membres décrits ne possèdent pas une représentation complexe de l’environnement, de ce fait, nous n’utilisons pas les langages RDFS/OWL ou les ACL[...]”.
- la modélisation des comportements en utilisant les langages PLOOM-UNITY qui sont des langages formels orientés agents.

La phase “implémentation”

Scoop propose deux façons de modéliser les systèmes coopératifs, soit grâce à des réseaux de Petri stochastiques, soit en utilisant un simulateur multi-agents.

Les réseaux de Petri stochastiques permettent de modéliser les interactions du système. La simulation de ces réseaux permet d’obtenir des résultats sur ces interactions.

La deuxième proposition qui nous intéresse d’avantage est la simulation par un système multi-agents. Le paradigme multi-agents étant conçu pour la modélisation, la simulation du travail collaboratif nous paraît plus appropriée.

Saint-voirin a appliqué sa méthode à la e-maintenance sur le démonstrateur PROTEUS. Les résultats qu'il a obtenus montrent un taux d'échec des diagnostics de panne de 77%. Ces résultats sont en accord avec les simulations qu'il a effectuées. En effet, il précise que le système PROTEUS ne dispose que d'un pool de 5 experts disponibles et d'un groupe de 3 experts. Or, ses simulations montrent que la taille idéale de ce pool serait d'environ une trentaine d'experts, la taille du groupe n'influençant que peu les résultats.

3.2.2 Les systèmes collaboratifs

Plusieurs exemples existent dans la littérature pour faire collaborer les composants des plates-formes de e-maintenance.

[Trappey et al., 2008] propose une chaîne de maintenance collaborative comprenant quatre étapes / modules principales : surveillance de l'état des équipements, pronostics et diagnostics, aide à la décision concernant la maintenance, ordonnancement. La collaboration dans et entre les modules est assurée par des agents. Il propose également une architecture générique collaborative entre les agents (cf. 5 page 23). Ce modèle a été implémenté avec le système multi-agents JADE. Le but de cet article est de donner les lignes directrices d'une structure relationnelle inter-agents pour l'implémentation d'une plate-forme de e-maintenance.

Un autre système à base d'agents a été proposé par [Yu et al., 2003] qui repose sur le système POMAESS (Problem-Oriented Multi-Agent based E-Service System). Trois agents sont identifiés (agent de production, maintenance, contrôle) qui répondent à un besoin spécifique : faire fonctionner une valve d'arrivée d'eau dans une centrale hydroélectrique. Ces travaux présentent une solution qui est trop dépendante du contexte pour qu'elle soit intéressante dans nos travaux.

Les travaux de [Seguy, 2008] étudient la décision collaborative dans les systèmes distribués et leur application à la e-maintenance. Ses travaux s'intéressent particulièrement à la communication et à la collaboration à travers les Technologies de l'Information et de la Communication (TIC).

Outre le travail de modélisation des processus de maintenance, des processus de collaboration, et des performances théoriques des systèmes de e-maintenance, les résultats des travaux de Seguy concernant la collaboration des acteurs de la maintenance et des TIC sont importants.

Ils montrent, par exemple, grâce à une simulation utilisant des réseaux de Petri stochastiques, une corrélation entre le taux de disponibilité des TIC

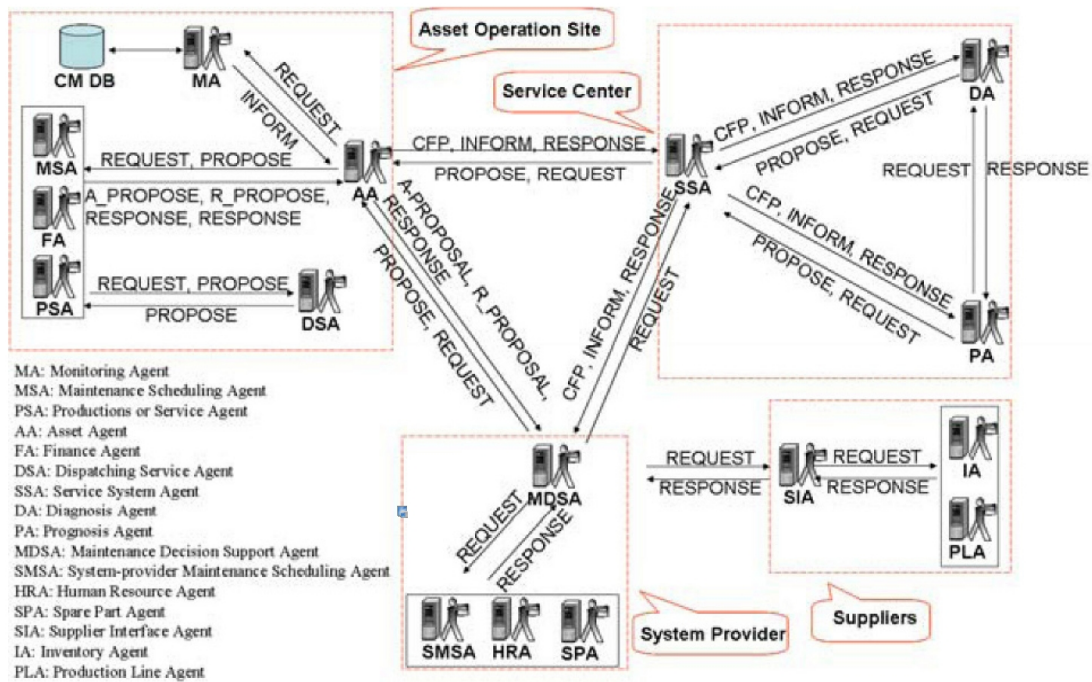


FIGURE 5 – Relation entre les agents d'un système générique de e-maintenance [Trappey et al., 2008]

dans les systèmes de e-maintenance et les temps de traitement des pannes. Plus la disponibilité des TIC est grande plus les temps de traitement (réparations) sont courts. Par exemple, pour une indisponibilité des TIC de 50%, la durée moyenne de traitement augmente également de 50%. La disponibilité des acteurs est également un facteur de rapidité des interventions indépendamment de la disponibilité des TIC.

La forte externalisation des ressources (expert distant) rend le processus de maintenance très dépendant des TIC lors de la phase collaborative. En d'autres termes, la collaboration entre les acteurs de la maintenance ne peut se faire que lors de communications. Ces communications passant par les TIC, elles dépendent fortement du bon fonctionnement de celles-ci.

En annexe de ses travaux, Seguy propose une modélisation Agent des systèmes collaboratifs de e-maintenance. Elle s'appuie sur la définition d'agents cognitifs et utilise l'outil MadKit.

4 Synthèse

Nos recherches sur la e-maintenance nous ont permis de nous familiariser avec le monde de la maintenance industrielle et de mesurer les avantages des plates-formes de maintenance. Nous avons ainsi pu appréhender les aspects distribués et coopératifs des plates-formes de e-maintenance.

Nous avons ensuite cherché à comprendre les enjeux de la sémantique dans les applications de maintenance et dans les web services. Cette sémantique est selon nous une des bases sur lesquelles la collaboration intelligente entre les éléments de la plate-forme et les humains peut avoir lieu. C'est également sur la base de ces informations enrichies grâce à une ontologie que les modules intelligents de la plate-forme pourront fournir des résultats plus pertinents.

Les recherches menées sur les systèmes coopératifs et collaboratifs nous ont permis de mieux intégrer les concepts de coopération et de collaboration. Nous avons également découvert des pistes intéressantes pour la modélisation de notre simulateur dans les travaux de Saint-Voirin et de Trappey.

Cet état de l'art concernant la maintenance, la sémantique et la collaboration avait également pour but de donner les bases nécessaires à la compréhension du projet SMAC que nous présentons dans la partie suivante.

Troisième partie

Présentation du projet SMAC

L’acronyme SMAC signifie S-MAintenance et Cycle de vie. Ce projet est réalisé en collaboration avec plusieurs acteurs universitaires

- l’AS2M du FEMTO-ST,
- le LIFC de l’université de Franche Comté,
- le LICP de l’EPFL,

et industriels

- le GIM-CH,
- TORNOS,
- Em@systec.

Nous allons présenter dans un premier temps le projet SMAC dans son aspect théorique en mettant l’accent sur les points forts qui en font un projet innovant. Dans un deuxième temps nous détaillerons plus avant le système cible d’un point de vue plus “physique”.

5 Le projet SMAC

Le projet SMAC a pour but le développement d’une plate-forme de s-maintenance se basant sur la plate-forme de e-maintenance développée dans le projet PROTEUS.

Nous avons vu la définition de la s-maintenance par Rasovska dans la section 1.2.2. Dans cette définition on remarque que les atouts de la s-maintenance sont l’interopérabilité sémantique des équipements et l’utilisation d’une ontologie afin de soutenir cette interopérabilité. Les avantages du projet SMAC ne s’arrêtent pas ici, ce projet a pour but de faciliter toutes les formes de collaborations au sein de la plate-forme de s-maintenance en utilisant les dernières innovations technologiques tel que la réalité augmentée. Un des derniers aspects importants du projet est la gestion du cycle de vie des produits au travers de méthodes de capitalisation des connaissances et de PLM (Product Life Management).

5.1 L’interopérabilité sémantique

Selon [Wikipédia, 2010] “l’interopérabilité est la capacité que possède un système à fonctionner avec d’autres produits ou systèmes informatiques, existants ou futurs, sans restriction d’accès ou de mise en œuvre”. en d’autres

termes c'est la faculté qu'ont plusieurs systèmes à s'interfacer afin de communiquer. Pour [Maedche and Staab, 2000] et [Halevy et al., 2005] l'interopérabilité des systèmes de maintenance est réalisée par des mécanismes intermédiaires comme par exemple des agents médiateurs. Ces mécanismes doivent toutefois avoir une connaissance spécifique du domaine pour coordonner différentes sources de données via la plupart du temps des ontologies.

L'interface entre les systèmes hétérogènes se fera donc ici au niveau du sens des informations transportées, chaque "agent" du système ayant accès à l'ontologie afin de comprendre ce sens.

5.2 L'ontologie et la capitalisation des connaissances

L'ontologie est au cœur de deux aspects importants de SMAC. Le premier est l'aspect "interopérabilité" détaillé au point précédant ; le second est l'aspect de "gestion du cycle de vie" et "capitalisation des connaissances" (cf. 1.2.1 page 14). En effet l'ontologie de SMAC a été structurellement pensée pour la gestion du cycle de vie des produits depuis leur conception à leur destruction ou recyclage (cf. 9.1 page 33). Chaque fois qu'une alerte est donnée sur l'état de santé d'un équipement le système enregistre les actions menées suite à cette alarme, les pièces de rechange utilisées, le personnel qui s'est chargé de l'opération de maintenance, etc.

5.3 La réalité augmentée

La réalité augmentée est une technologie qui permet de superposer à la réalité, généralement à l'aide de lunettes spécialement conçues à cet effet, des informations utiles pour l'utilisateur de cette technologie. Un exemple des possibilités de cette technologie est illustré par la figure 6. Dans SMAC la réalité augmentée peut servir à visualiser une panne identifiée par le système, suivre des indications données en temps réel par un expert distant, etc. Ce dispositif prometteur peut ainsi faciliter la collaboration des hommes et des machines par un affichage temps réel des informations nécessaires à un fonctionnement optimal du système de maintenance.

5.4 La collaboration

La collaboration est présente au sein du système à plusieurs niveaux. Comme expliqué dans le point précédent, elle est favorisée pour les humains par l'utilisation de la réalité augmentée. Pour les agents logiciels la collaboration peut être mise en place grâce à l'interopérabilité sémantique garantissant une bonne compréhension lors des échanges inter-machines.

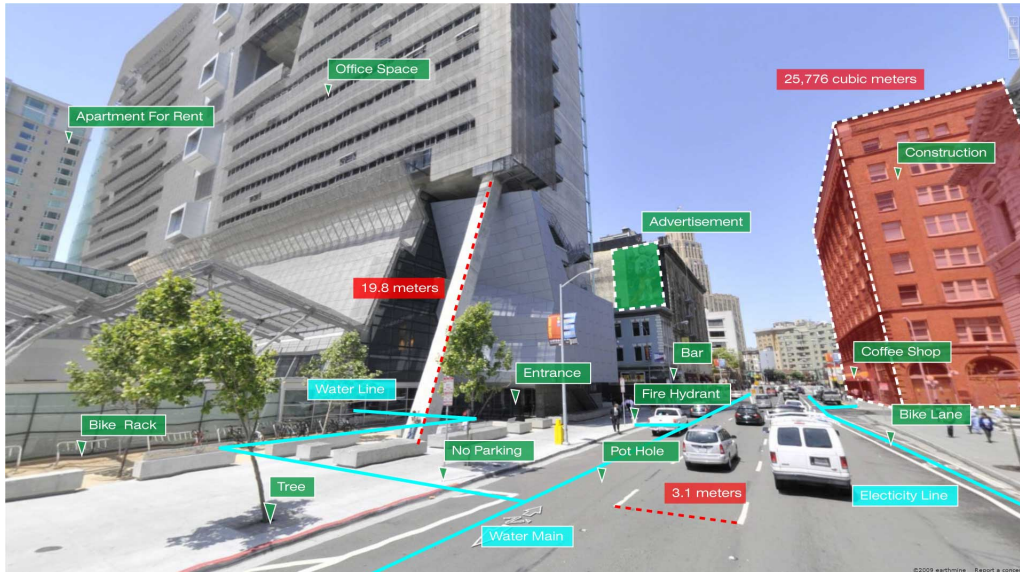


FIGURE 6 – Exemple de réalité augmentée

5.5 Conclusion

Le projet SMAC est le premier et le seul (à notre connaissance) qui tente de développer une plate-forme de maintenance sémantique. Il est basé en partie sur les travaux d'Ivana Rasovska [Rasovska, 2006] [Rasovska et al., 2007] pour les aspects sémantiques et de David Saint-Voirin [Saint-Voirin, 2006] pour les aspects collaboratifs. Ce projet innove sur plusieurs points, les plus importants étant l'interopérabilité sémantique et l'utilisation de la réalité augmentée en sus des autres médiums de communication.

6 Système cible

SMAC a pour but d'apporter une approche théorique et pratique à la s-maintenance. Les principes et les méthodes proposés dans SMAC font l'objet d'une implémentation sur un système existant hérité du projet PROTEUS. La figure 7 montre les interactions possibles dans la plate-forme entre les composants que nous allons présenter ici.

Em@Web

Selon le site de la société Em@systec, Em@Web est une plate-forme informatique intégrée, avec un portail Web qui lui sert de point d'entrée, capable de répondre à n'importe quelle stratégie d'e-maintenance en s'appuyant sur

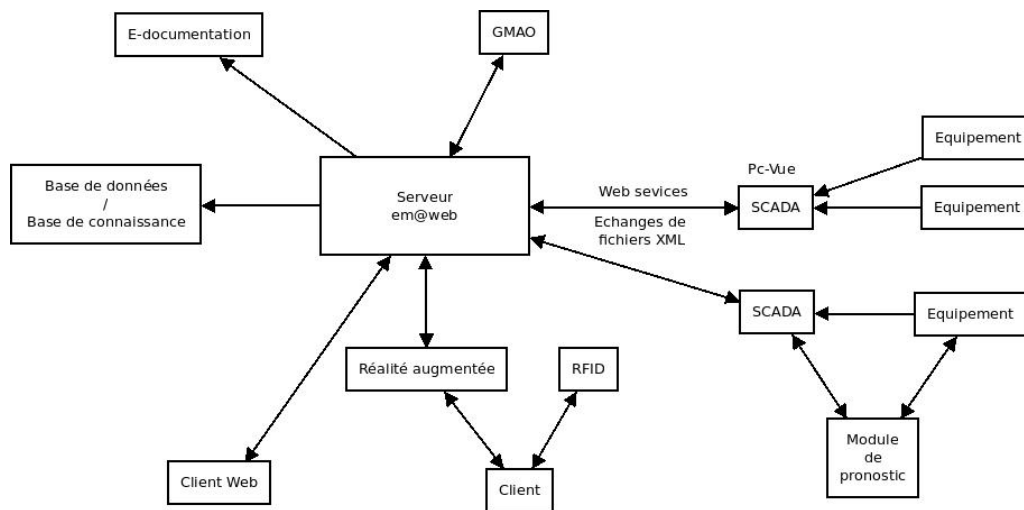


FIGURE 7 – Le système cible

des formats standardisés et non-propriétaires. Le portail accessible à tous les métiers concernés par la maintenance se connecte à l'ensemble des outils logiciels de maintenance et ainsi regroupe l'ensemble des données techniques, financières et logistiques d'un équipement ou d'une installation industrielle.

Dans notre cas, ce logiciel est le centre névralgique de la plate-forme de s-maintenance. C'est lui qui est en mesure de générer des diagnostics en analysant les données reçues des SCADAs.

GMAO

GMAO est l'acronyme de Gestion de Maintenance Assistée par Ordinateur. C'est l'outil qui permet de gérer la traçabilité des actions de maintenance. La GMAO est renseignée par les techniciens de maintenance à chaque fois qu'ils effectuent une opération de maintenance.

Base de données

La base de données est en relation directe avec Em@Web. Elle représente la base de connaissances du système.

E-documentation

La E-documentation est une autre source de connaissance. Elle est plus destinée aux acteurs humains qu'informatiques. Elle est consultée par les techniciens ou les experts de la maintenance pour mener à bien leurs tâches.

SCADA

SCADA est l'acronyme de Supervisory Control And Data Acquisition (télésurveillance et acquisition de données). Le SCADA est l'élément du système qui récupère et transmet les données recueillies par les capteurs sur les équipements ou par le module de pronostic.

Équipements

Les équipements sont les machines qui sont sur le site de production. Elles sont munies de capteurs qui recueillent et transmettent les informations relatives à leur état.

Pronostic

Un module de pronostic est ajouté à chaque équipement. Il communique avec les capteurs qui lui envoient des informations et avec le SCADA à qui il envoie ses pronostics.

Client

Les clients de la plate-forme (comprenez client au sens de client-serveur) sont de deux types : un client web classique et un client utilisant le système de réalité augmentée.

Client web

Le client web est un navigateur internet placé sur un terminal quelconque. Le service permettant d'accéder au portail de la plate-forme est géré par le logiciel Em@Web.

Réalité augmentée

Le client de réalité augmentée sera un terminal portable de type PDA qui pourra être complété de lunettes intégrant un écran à cristaux liquides permettant de recevoir en temps réel les informations de la plate-forme.

7 Synthèse

Nous avons vu que le projet SMAC est un projet ambitieux qui regroupe de nombreux acteurs de la maintenance à un niveau international. Il est la suite logique du projet européen PROTEUS sur la e-maintenance. La principale nouveauté du projet est l'utilisation d'une ontologie pour intégrer l'interopérabilité entre les différentes parties du système à un niveau sémantique, cette interopérabilité ayant pour conséquence le développement du travail

collaboratif avec et au sein de la plate-forme. Nous avons également détaillé le système cible qui est l’instanciation du modèle théorique de SMAC.

La section suivante présente le travail de modélisation de la plate-forme qui est au croisement entre les aspects théoriques et pratiques de SMAC.

Quatrième partie

Vers une modélisation agent de la s-maintenance

La contribution théorique de ce stage consistait à modéliser et implémenter la plate-forme SMAC. Nous avons dû pour ce faire, choisir judicieusement le paradigme de programmation pour que la simulation soit le plus proche possible du système réel.

Dans un premier temps nous justifierons le choix du paradigme multi-agents pour la réalisation de la simulation. Dans un deuxième temps nous présenterons les composantes sémantiques dans les communications entre les agents, ce qui est nous le rappelons, l'intérêt majeur de cette simulation. Dans une troisième partie nous donnerons les clés nécessaires à la compréhension du modèle en présentant le langage AML ainsi que les contraintes et modifications apportées à ce langage pour la modélisation de la plate-forme. Dans une quatrième et dernière partie nous exposerons le modèle que nous avons développé ainsi que la relation qu'il entretient avec l'ontologie. Enfin, nous conclurons cette partie par une synthèse de cette contribution.

8 Choix du paradigme Agent

Nous avons fait le choix d'utiliser le paradigme multi-agents pour la modélisation et la simulation de la plate-forme SMAC. Ce choix nous est apparu comme le plus naturel, étant donné la forte distribution du système (cf. figure 7 page 29) ainsi que la prédominance de la collaboration entre les acteurs de ce système (cf 5.4 page 27). En effet, le paradigme multi-agents permet de voir un système comme un ensemble de systèmes autonomes capables de perception et d'actions (les agents) dans un environnement. Ces agents sont liés par des relations à d'autres agents et aux objets de leur environnement [Ferber, 1995]. D'après cette définition nous avons donc identifié ce paradigme de programmation comme étant le plus proche du système logique et physique de SMAC. Nous avons ainsi pu modéliser de façon satisfaisante le système et les interactions en son sein.

Un autre aspect important qui a été à la base du choix d'utilisation des systèmes multi-agents est la modélisation des humains agissant sur le système. Les agents de par leur nature autonome se prêtent très bien à une modélisation anthropomorphique des acteurs du système ; de plus, les travaux

de [Saint-Voirin, 2006] avaient déjà exploré cette voie pour la e-maintenance. Nous nous sommes donc inspiré de ces travaux pour mener à bien les nôtres.

Enfin, la nature sémantique (détaillée dans la section 9.1 page 33) et interopérable des éléments du projet promettait une bonne utilisation des systèmes multi-agents grâce notamment aux recommandations de la *Foundation for Intelligent Physical Agents* (cf 9.2.1 page 36) détaillées dans les sections suivantes.

9 Sémantique dans les échanges inter-agent

Les agents sont des entités communicantes. Il y a principalement deux modes de communication pour un agent : la consultation et l'écriture sur un "tableau d'affichage" et l'envoi et la réception de messages [Ferber, 1995]. Les agents représentent/simulent les acteurs du système aussi bien humains que non-humains. Les acteurs non-humains doivent assurer une interopérabilité sémantique, c'est à dire qu'ils doivent communiquer sur les concepts qu'ils manipulent et dans une certaine mesure les "comprendre".

A cette fin, le projet SMAC a développé une ontologie de maintenance en prenant appui sur le modèle PROMISE et le projet PROTEUS [Aristeidis, 2010].

Dans une première section nous présenterons l'ontologie de SMAC dans ses grandes lignes puis nous ferons un rapide tour d'horizon des recommandations de la *Foundation for Intelligent Physical Agents* (FIPA) pour l'utilisation d'ontologies dans la communication entre agent.

9.1 L'ontologie SMAC

Le modèle de l'ontologie SMAC est tiré des travaux du projet PROMISE et du projet PROTEUS [Aristeidis, 2010]. Le projet PROMISE a développé un modèle ontologique de gestion du cycle de vie des produits [Maurizio Tomasella, 2006]. Selon le site web du projet, PROTEUS "est un projet européen qui a rassemblé une quinzaine de partenaires européens dans le but de réaliser une plate-forme logicielle générique pour implanter des centres de e-maintenance en utilisant les Web services".

L'ontologie du projet SMAC a fusionné le modèle développé pour PROTEUS avec celui de PROMISE. Le modèle SMAC est centré sur le "produit" et son cycle de vie.

Nous allons maintenant expliquer le modèle (cf. Figure 8 Pour une compréhension facile il est conseillé de se reporter à la partie 6 page 28). Le

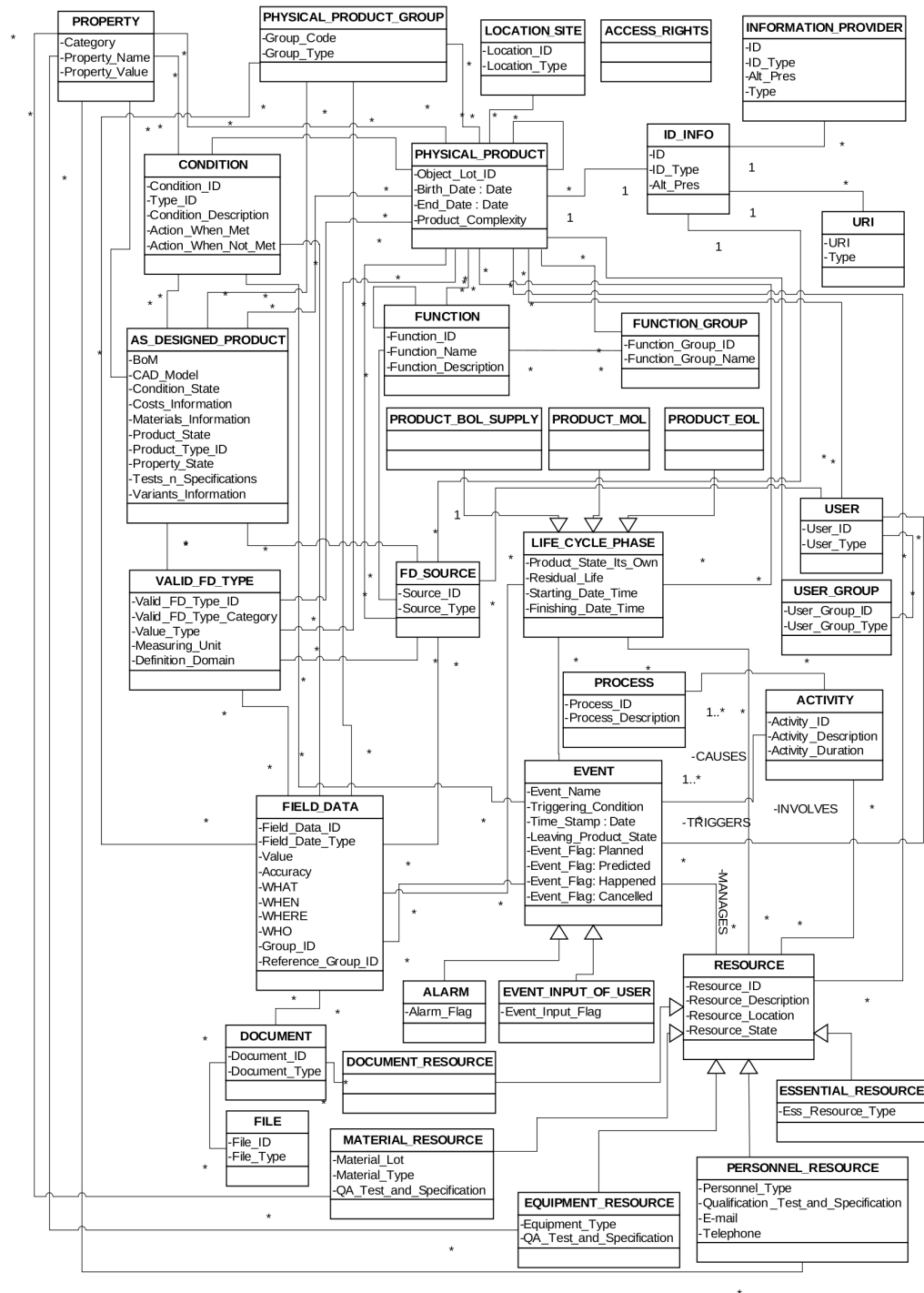


FIGURE 8 – L'ontologie de SMAC [Aristeidis, 2010]

modèle s’articule autour de trois points que nous détaillerons ci-après :

- Le produit, pour nous il s’agit des équipements (AS_DESIGNED_PRODUCT et PHYSICAL_PRODUCT),
- le cycle de vie (LIFE_CYCLE_PHASE),
- les actions entreprises pour maintenir le produit dans un état satisfaisant (ACTIVITY).

9.1.1 Le produit

La classe AS_DESIGNED_PRODUCT représente le produit ou l’équipement neuf, à sa sortie d’usine. Cette classe est liée à la classe CONDITION qui enregistre toutes les conditions normales de fonctionnement du produit ainsi que les actions à entreprendre lorsque ces conditions ne sont pas rencontrées. La classe AS_DESIGNED_PRODUCT correspond à la phase BOL(Beginning Of Life) du cycle de vie.

La classe PHYSICAL_PRODUCT représente le produit tout au long de sa vie. Elle est liée à la classe AS_DESIGNED_PRODUCT pour garder les références et conditions du produit.

La classe FIELD_DATA est liée à la classe PHYSICAL_PRODUCT. Elle représente les valeurs instantanées qui sont enregistrées pour chaque produit. C’est pour nous les valeurs envoyées par les SCADAs, les SCADAs étant eux de la classe FD_SOURCE.

9.1.2 Le cycle de vie

Toutes les instances de FIELD_DATA sont liées à une instance de la classe LIFE_CYCLE_PHASE ou une de ses classes fille. Cette classe enregistre tous les événements EVENT déclenchés par un franchissement de seuil des conditions. Lorsqu’un événement survient, s’il est validé, il entraîne une activité ACTIVITY (cf. section 9.1.3). Ainsi, grâce aux liens entre le produit, les événements, les activités et le cycle de vie, les acteurs et manageurs peuvent avoir une bonne traçabilité du produit et des actions qui y ont été opérées.

9.1.3 Les actions

Les actions ACTIVITY sont toujours déclenchées par des événements ; elles ont besoin de ressources pour être menées à bien que ce soit des pièces de rechange MATERIAL_RESOURCE, des outils EQUIPMENT_RESOURCE, ou des ressources humaines PERSONNEL_RESOURCE. Les activités peuvent

également dépendre d'un certain nombre de documents nécessaires à la réalisation de l'activité (par exemple une notice de démontage).

9.1.4 Conclusion

L'ontologie de SMAC permet aux agents de manipuler les concepts et les idées à travers un réseau de classes. Elle est également la garante de la traçabilité des produits et des actions entreprises pour prolonger leur durée de vie. Elle permet également aux agents d'avoir "conscience" d'eux-mêmes et des autres en tant que concepts (cf 11.3 page 51.)

Afin d'utiliser au mieux cette ontologie, nous avons besoin de pouvoir organiser et structurer les messages qui y feraient référence, nous avons donc choisi d'utiliser les spécifications de la FIPA que nous détaillons dans la section suivante.

9.2 Les spécifications de la FIPA

Nous nous sommes intéressé aux travaux de la FIPA premièrement pour la gestion des ontologies que proposait l'ACL (Agent Communication Language). Nous nous sommes ensuite rendu compte que cette spécification proposait un panel exhaustif de gestion des actes de langage ce qui était très intéressant pour travailler de façon structurée sur les échanges inter-agents.

Nous présenterons dans cette section les principaux intérêts que nous avons trouvés aux spécifications de la FIPA. Nous ferons premièrement une présentation de la spécification FIPA-ACL. Nous continuerons par détailler les paramètres de messages que nous jugeons importants. Enfin, nous finirons par une rapide présentation de la spécification FIPA-OSS (Ontology Service Specification).

9.2.1 La spécification FIPA-ACL

La spécification FIPA-ACL est une spécification qui détermine la structure des messages envoyés et reçus par les agents. L'un des avantages de cette structure est que toutes les informations pour l'envoi, la réception et la compréhension du message (ou tout du moins l'endroit où les trouver) sont enregistrées dans le message. Le tableau 1 montre les différents paramètres disponibles pour un message. La section suivante détaillera les paramètres les plus importants et utiles dans l'utilisation de cette spécification.

| Parameter | Category of Parameters |
|-----------------|------------------------------|
| performative | Type of communicative acts |
| sender | Participant in communication |
| receiver | Participant in communication |
| reply-to | Participant in communication |
| content | Content of message |
| language | Description of Content |
| encoding | Description of Content |
| ontology | Description of Content |
| protocol | Control of conversation |
| conversation-id | Control of conversation |
| reply-with | Control of conversation |
| in-reply-to | Control of conversation |
| reply-by | Control of conversation |

TABLE 1 – FIPA ACL Message Parameters [FIPA, 2002]

9.2.2 Les paramètres des message

Nous allons détailler ici les paramètres importants pour notre cas.

Performative

Le paramètre “performative” est très important, c’est dans celui-ci qu’on définit le type d’acte de langage pour lequel le message est envoyé. “Les actes de langage désignent l’ensemble des actions intentionnelles effectuées au cours d’une communication” [Ferber, 1995], c’est à dire les actions engendrées par la communication. Un exemple simple : si on demande à son voisin de table de nous passer le sel, on ne s’intéresse pas à la valeur de vérité de cette phrase mais à l’action qui va lui succéder. Une performative peut être par exemple “Inform” (envoi d’une information), “Request” (envoi d’une demande d’action), “Accept Proposal” (accepte de réaliser l’action demandée) ou encore “Confirm” (confirmation qu’une proposition est vraie).

Sender et Receiver

Ils sont les identités de l’émetteur de l’acte de langage (sender) et du destinataire (receiver).

Language

Ce paramètre représente le langage utilisé dans le message, un agent pouvant tout à fait être multilingue. Ce paramètre peut être renseigné par des

langages naturels tel le français ou l'anglais mais aussi des langages informatiques comme FIPA SL, XML ou encore RDF. Dans notre cas nous utiliserons RDF (cf. partie V page 54).

Ontology

L'ontologie est le modèle de représentation des connaissances dont l'agent a besoin pour comprendre le sens du message(cf. section 9.1 page 33).

9.2.3 La spécification FIPA-OSS

La spécification OSS de la FIPA est encore au stade expérimental (c'est à dire que la fondation n'a pas constaté assez d'implémentation de cette spécification pour la considérer comme un standard)[FIPA, 2001]. Elle a été utile pour nos travaux surtout pour la définition d'un OA (OntologyAgent). Nous n'avons pas suivi à la lettre la spécification mais nous nous en sommes inspiré pour créer un agent de management de l'ontologie.

Cette spécification définit les services que doit fournir un OA :

- Aider les agents à choisir une ontologie pour la communication
- Créer et mettre à jour les ontologies
- Traduire des expressions entre différentes ontologies
- Répondre aux requêtes concernant les relations entre les termes ou les ontologies
- Découvrir des ontologies publiques afin de pouvoir y accéder

Notre système ne comportant qu'une seule ontologie, nous ne nous sommes intéressés qu'aux deuxième et quatrième points. Ainsi, notre OA n'est chargé que de la gestion de l'ontologie et de la diffusion de cette ontologie aux agents qui souhaitent l'utiliser. Pour plus de détails, il faut se référer à la partie V page 54.

9.3 Conclusion

Nous avons vu que la sémantique dans les échanges inter-agent est indispensable à la simulation de la plate-forme SMAC étant donné le choix du paradigme multi-agents pour la simulation et l'utilisation de l'ontologie développée dans le projet. Pour structurer de manière efficace les échanges de messages enrichis d'une sémantique nous nous sommes intéressé aux spécifications de la FIPA ; deux spécifications ayant particulièrement retenu notre attention : FIPA-ACL et FIPA-OSS. Elles permettent de structurer les échanges de messages de manière efficace et "tout en un", toutes les informations étant contenues dans le message. Après avoir déterminé la façon

dont structurer les échanges sémantiques il nous fallait trouver une façon de structurer les agents, ce travail est présenté dans la section suivante.

10 AML et systèmes complexes

Trouver la bonne structuration des agents au sein de la simulation a été un objectif important. Pour le réaliser nous nous sommes inspiré des travaux de Thibaud Brocard sur les systèmes complexes [Brocard, 2010]. Avant de présenter notre modèle (section 11) nous allons, dans une première section, donner les clés du langage AML (Agent Modeling Language) [Whitestein, 2004] avec lequel a été réalisé la modélisation des systèmes complexes puis dans une deuxième section présenter le travail mené sur le sujet.

10.1 Agent Modeling Language - AML

“AML est un langage de modélisation visuel semi-formel. Il sert à modéliser, spécifier et documenter les systèmes qui incorporent les concepts de la théorie des systèmes multi-agents.” [Whitestein, 2004]

AML est un langage très vaste et ouvert qui permet de modéliser toutes les notions abordables dans un système multi-agents, nous n’allons donc présenter ici que les concepts utiles à la compréhension de notre modélisation. AML est un langage dérivé d’UML. Ainsi, un diagramme de structure AML peut se voir comme un diagramme de classe particulier. A la différence de ceux-ci, les “classes” sont représentées par un pictogramme qui peut être seul (exemple fig.16) ou dans un cadre comme par exemple sur la figure 9.

10.1.1 Concepts Structurels

Nous présentons ici les différents concepts qui définissent le contenu du système, c’est à dire les agents et leurs ressources. On peut remarquer que le concept “environnement” n’est pas présenté, en effet dans notre simulation (cf. 11 page 45) l’environnement de la simulation sont les sites et ils sont considérés comme des groupes (cf. figure 17).

Type d’agent AML

Un type d’agent (fig.9) est proche du concept de classe d’objet. Ainsi, les agents du système sont des instances de ces types d’agent (c’est également vrai pour les autre types). Chaque agent possède une liste d’attributs et d’opérations comme n’importe quel objet. Le formalisme AML nous permet également de définir un agent comme étant la composition de plusieurs autres


| | |
|--------------------------|---|
| <<agent>> Name |  |
| <i>attribute list</i> | |
| <i>operation list</i> | |
| <i>parts</i> | |
| <i>behaviors</i> | |

FIGURE 9 – Type d’agent AML

agents. Cette possibilité ne sera pas exploitée conformément aux travaux de T.Brocard. Un agent possède également un ou plusieurs comportements qui représentent sa façon de réagir aux événements qui se produisent dans le système et aux actes de langage qu’il reçoit des autres agents.


| | |
|-----------------------------|---|
| <<resource>> Name |  |
| <i>attribute list</i> | |
| <i>operation list</i> | |
| <i>parts</i> | |
| <i>behaviors</i> | |

FIGURE 10 – Type de ressource AML

Type de ressource AML

Une ressource (fig.10) dispose des mêmes champs qu’un agent. La différence vient du fait que contrairement à un agent une ressource n’est pas une entité autonome, il ne s’agit que d’un objet que les agents utilisent. Dans notre cas, nous n’utiliserons donc pas de comportement pour une ressource.

10.1.2 Concepts sociaux

Nous présentons ici les différents concepts liés à l’organisation et aux relations entre les agents.

Type d’unité organisationnelle AML

Une unité organisationnelle (fig.11) est typiquement composée de plusieurs agents agissant à l’intérieur de cette unité conformément à l’organisation mise

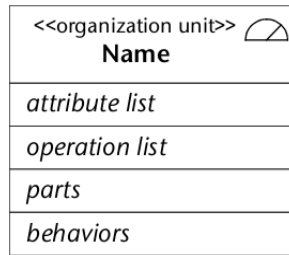


FIGURE 11 – Type d’unité organisationnelle AML

en place au sein de l’unité. Ce concept sera utilisé pour modéliser les groupes du paradigme AGR.

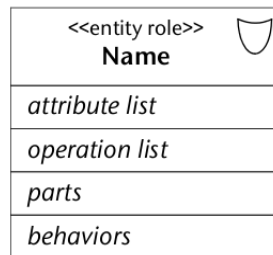


FIGURE 12 – Type de rôle AML

Type de rôle AML

Chaque agent est susceptible de jouer un ou plusieurs rôles (fig.12) au sein d’une organisation. Un agent peut changer de comportement suivant les rôles qu’il joue.

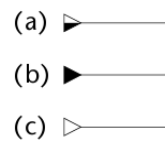


FIGURE 13 – Propriétés Sociales AML

Propriétés Sociales AML

Suivant les rôles joués, un agent sera le pair, le subordonné ou le superior-

donné d'autres agents. Ces relations sont modélisées comme sur la figure 13 tel que :

- (a) est une relation de pair
- (b) est une relation de superordonné
- (a) est une relation de subordonné


| |
|---|
| <<behavior fragment>>  |
| Name |
| <i>attribute list</i> |
| <i>operation list</i> |
| <i>parts</i> |
| <i>behaviors</i> |

FIGURE 14 – Fragment de comportement AML

Fragment de comportement AML

Un fragment de comportement (fig.14) est soit un comportement réutilisable par plusieurs agents, soit une collection d'actions simples que peut réaliser un agent, c'est à dire une action complexe. Par exemple pour un agent humain "Resserrer une vis" est une action simple qui peut être intégrée dans un fragment de comportement qui serait "Réparer une machine".

10.2 Modélisation agent des systèmes complexes.

Selon [Brocard, 2010] "on considère un système comme complexe lorsqu'on ne peut plus prévoir son comportement global soit à cause du nombre élevé d'entités qui le compose, soit à cause de leur cognition trop élevée". La méthode de représentation de ces systèmes mise au point par T.Brocard restreint le formalisme AML notamment en incluant les concepts AGR définis par [Ferber, 1995]. Nous expliquerons ces restrictions dans la première partie de cette section. La méthode à proprement parler décompose les systèmes complexes en deux parties sur plusieurs niveaux pour créer un modèle de structure pour les systèmes multi-agents. Cette structure est basée sur une organisation holarchique des agents et de groupes agentifiés ; c'est-à-dire qu'un groupe d'agents peut être considéré comme un agent à part entière. Ce modèle sera présenté dans la deuxième partie.

10.2.1 Les restrictions d'AML

Le méta-modèle de la figure 15 montre les concepts qui sont modélisés dans le modèle de représentation des systèmes complexes. On peut ainsi voir que le modèle s'appuie sur des agents, des groupes et des rôles. En effet ce modèle s'appuie sur le paradigme AGR. Nous allons détailler ici les restrictions/modifications apportées à AML pour spécifier ce modèle.

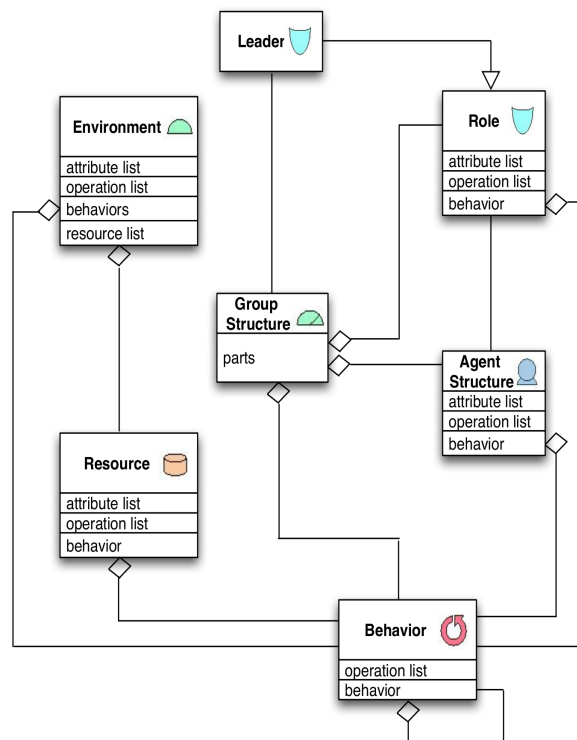


FIGURE 15 – Méta-modèle de structure des systèmes complexes [Brocard, 2010]

Les agents

Dans le modèle, les types d'agents sont nommés "AStruct" ; ils sont définis dans la partie organisationnelle du modèle de structure (cf.10.2.2). Tous les agents du système sont donc des instances de ces "AStruct". La partie "parts" de l'agent restera toujours vide, T.Brocard considérant qu'un agent ne peut être composé d'autres agents. La partie "behaviors" comportera les fragments de comportement modélisant des actions complexes.

Les groupes

En AML, la notion de groupe n'existe pas. Le concept le plus proche du groupe est le "type d'unité organisationnelle" ; c'est ce concept qui servira de support à la notion de groupe. " Dans le modèle, un groupe aura ses parties "attribute list", "operation list" et "behavior" vides. Seule la partie "parts" sera remplie : elle permet de décrire la structure du groupe, c'est-à-dire de quoi il est composé. Un groupe peut être composé d'agents, d'autres groupes, voire des deux." [Brocard, 2010]

Les rôles

Les rôles représentés en AML sont assez proches de la notion de rôle en AGR, néanmoins, un rôle ne peut pas être composé d'autres rôles, la partie "parts" sera donc toujours vide. Un rôle de "leader" est toujours créé pour un groupe, c'est ce leader qui sera l'interface entre les autres agent et le reste du groupe. Dans notre modélisation ce rôle est implicite et n'apparaît pas sur les diagrammes.

L'environnement et les ressources

L'environnement est une entité globale du système. Néanmoins, cette entité n'est pas indispensable au système. Quand il est présent, les ressources lui sont rattachées.

"Les ressources sont des entités physiques (par exemple des matières premières dans un système de production) ou des entités informationnelles (par exemple une base de données)". La partie "parts" des ressources est toujours vide car elles sont considérées comme un tout.

Les actions complexes

Les actions complexes (cf. 10.1.2 page 42) sont modélisées par des fragments de comportement. "Si l'action complexe se divise en plusieurs actions simples, celles-ci apparaîtront dans la partie "operation list". Si l'action complexe se divise en plusieurs actions elles aussi complexes, elles apparaîtront dans la partie "behaviors"."

10.2.2 Le modèle de structure

Le modèle de structure est découpé en trois parties :

- l'aspect organisationnel
- l'aspect social
- l'instanciation

Nous ne nous sommes pas intéressés à l'instanciation du modèle ne souhaitant pas imposer une instanciation fixe de la simulation. Nous ne présenterons

donc que les deux premiers aspects. Chaque aspect est divisé en plusieurs niveaux d'abstraction, du niveau 0 au niveau X.

Aspect organisationnel

L'aspect organisationnel décrit l'agencement des agents les uns par rapport aux autres et leur environnement. Il comporte quatre types d'entités :

- les agents
- les groupes agentifiés
- l'environnement
- les ressources

Au niveau 0 apparaissent les agents, les ressources et l'environnement (s'il est présent). Au niveau 1 et supérieurs, les agents du niveau 0 sont utilisés pour la formation de groupes agentifiés. Enfin, le niveau maximum représente l'ensemble du système.

Aspect social

L'aspect social du modèle décrit les rôles et les liens hiérarchiques entre les agents. Cet aspect possède autant de niveaux que l'aspect organisationnel. Au niveau le plus bas sont décrit tous les rôles que pourront jouer les agents. A partir du niveau 1 les rôles sont donnés à des groupes ; c'est aussi à partir de ce niveau que les liens hiérarchiques sont définis au sein des groupes.

10.3 Conclusion

Le langage AML et les travaux de Thibaud Brocard sont deux outils bien adaptés à la modélisation des systèmes complexes en général et de la plateforme SMAC en particulier. Ils permettent de développer de façon claire et structurée un modèle facilement compréhensible par des non informaticiens tout en donnant assez de robustesse à l'ensemble pour la base d'une implémentation. La section suivante montrera l'utilisation que nous en avons faite.

11 Modèle SMA pour la simulation de la collaboration

Cette section présente le travail de modélisation effectué pour représenter le système SMAC sous une forme de système multi-agents. Nous commencerons par présenter le modèle organisationnel, puis le modèle social. Nous verrons ensuite les correspondances entre les classes définies dans l'ontologie

de SMAC et celles définies dans le modèle organisationnel. Nous récapitulons enfin le travail de modélisation dans la conclusion de cette section.

11.1 Structure organisationnelle

Comme présentée à la section précédente, la partie organisationnelle du modèle de structure définit les agents et les groupes de notre modèle.

11.1.1 Niveau 0

Au niveau 0 (voir fig.16) on trouve la spécification des agents humains et informatiques qui composent le système. On trouve aussi les ressources dont ils disposent pour l’accomplissement de leurs tâches. Les compétences d’un agent sont vues comme une ressource de cet agent. Les compétences d’un groupe agentifié “Unité de maintenance” sont l’ensemble des compétences des agents qui le composent. On trouve également ici les équipements avec leurs capteurs liés aux SCADAs.

11.1.2 Niveau 1

Le niveau 1 (voir fig.17) spécifie les groupes “Site” et “Unité de maintenance”. Un “Site” est un site au sens géographique du terme, il peut représenter une usine, un bureau, une partie de bâtiment, un atelier... Il n’est pas utilisable tel quel. Le Site est découpé en deux sous groupes le “Site de maintenance” et le “Site de production”. C’est un découpage logique, non géographique. Dans une petite structure, un site de maintenance et un site de production peuvent être au même endroit. Le site de production sera le site où se trouvent les équipements à maintenir, par exemple la pièce principale où sont les machines de production. Un site de maintenance sera un site où se trouvent les différents acteurs de la maintenance (humain comme logiciel). Ce peut être un atelier de réparation, la salle serveur où se trouvent les logiciels de la plate-forme, l’entreprise qui sous-traite une partie de la maintenance, etc.

Pour les tâches de maintenance à effectuer, les acteurs humains de la maintenance devront se déplacer sur le site de production. Une unité de maintenance est un groupe agentifié représentant les acteurs de la maintenance qui collaborent à l’exécution d’une tâche de maintenance. Une unité de maintenance peut ne contenir qu’un seul agent (mais toujours au moins un) s’il possède les compétences nécessaires pour la tâche qu’il a à réaliser. Par exemple, un technicien fait une inspection des équipements. Cette même

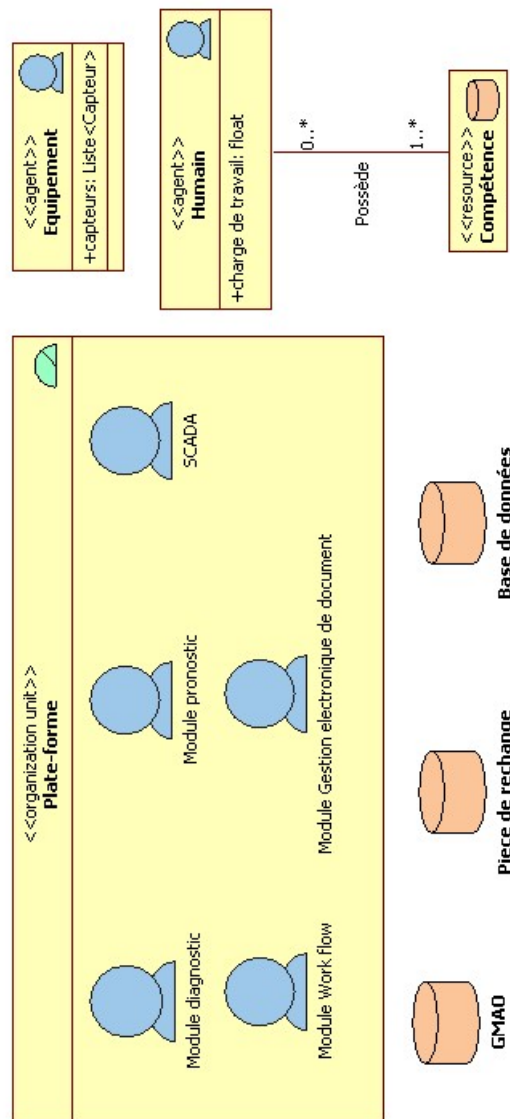


FIGURE 16 – Structure organisationnelle Niveau 0

unité de maintenance peut évoluer dynamiquement pour être composée de plusieurs agents (voir de plusieurs unités de maintenance si le problème est vraiment complexe) collaborant pour mener l'opération de maintenance à bien.

Exemple : Un technicien découvre une anomalie sur un équipement. Le problème sortant du cadre de ses compétences, il demande une procédure de maintenance à Em@Web. Em@Web a alors intégré l'unité de maintenance et collabore dorénavant avec le technicien. Em@Web propose de contacter un expert pour résoudre le problème. L'expert se trouve dans une autre entreprise (service technique du fournisseur de l'équipement) et communique avec le technicien pour le guider dans l'accomplissement de la réparation par l'utilisation de la réalité augmentée par exemple.

11.1.3 Niveau 2

Le niveau 2 représente le système de S-maintenance dans son ensemble. Comme on peut le voir sur la figure 18 le système est composé de sites et d'unités de maintenance. Ces deux groupes agentifiés sont ceux développés au niveau 1.

11.2 Structure sociale

La partie sociale du modèle décrit les rôles et les liens hiérarchiques entre les agents de la plate-forme SMAC.

11.2.1 Niveau 0

La plate-forme SMAC étant une plate-forme de maintenance, nous avons défini que le rôle de chaque agent du système, qu'il soit technicien, expert ou encore société de service, est un rôle de mainteneur, chaque agent ayant à charge la maintenance du système. Nous avons également défini des agents "équipement" (et leur rôle associé) dans le cas où se présenteraient des équipements dit "intelligents" avec des modules de diagnostic intégrés par exemple.

11.2.2 Niveau 1

Nous avons défini dans le niveau 1 que les acteurs de la maintenance, les mainteneurs, collaboraient entre eux aux sein d'une unité de maintenance pour effectuer les opérations de maintenance qui leur étaient attribuées.

11.2.3 Niveau 2

En vue de maintenir l'ensemble des équipements et la bonne marche du système, les unités de maintenance peuvent éventuellement collaborer entre elles si le besoin s'en fait sentir. La collaboration peut être implicite dans

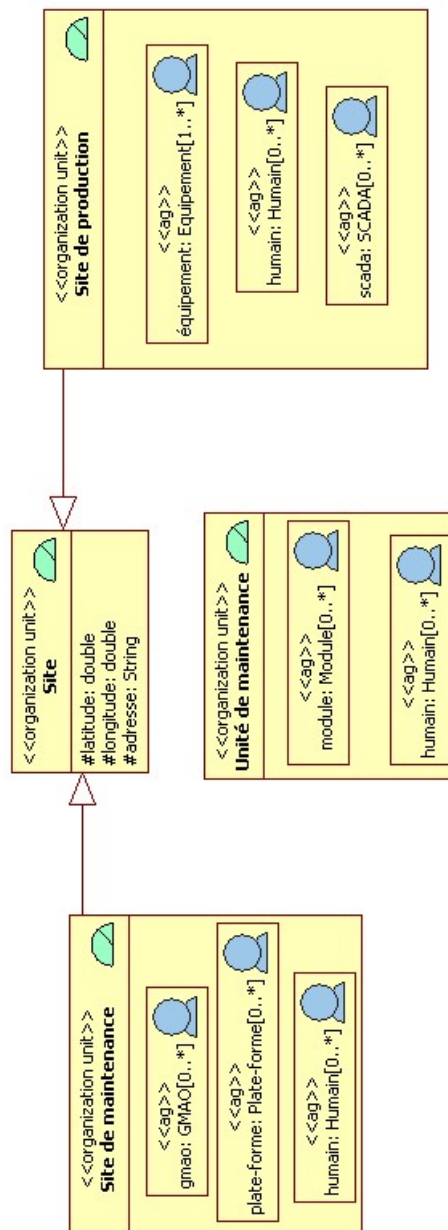


FIGURE 17 – Structure organisationnelle Niveau 1

le cas où le fait d'effectuer des tâches de maintenance sur une partie du système aidera une autre partie à fonctionner correctement. Elle peut être explicite pour un problème complexe à résoudre ; dans ce cas si deux unités

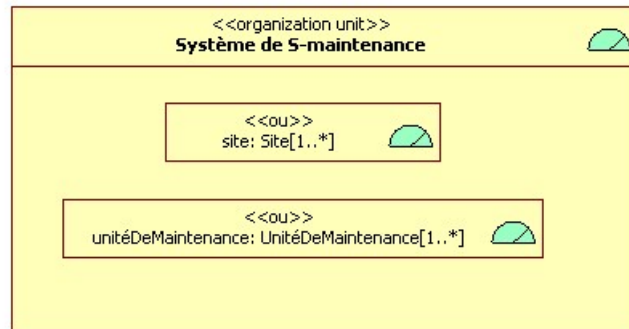


FIGURE 18 – Structure organisationnelle Niveau 2

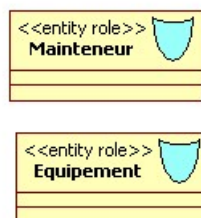


FIGURE 19 – Structure sociale niveau 0

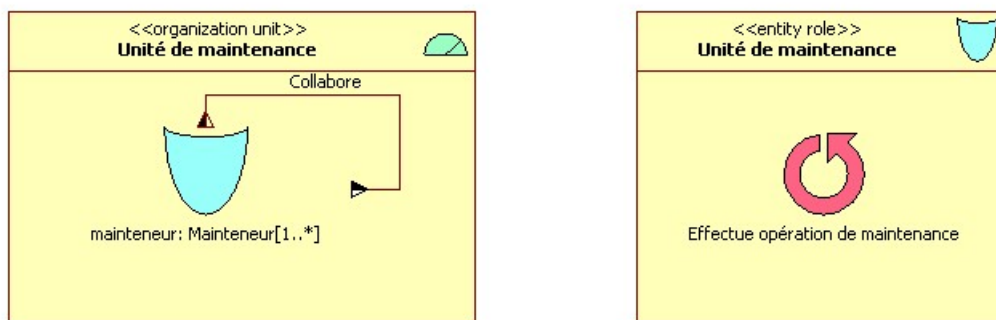


FIGURE 20 – Structure sociale niveau 1

de maintenance collaborent sur une opération précise en tant que groupe agentifié elles formeront une nouvelle unité de maintenance, la collaboration

| Agent | Classe ontologique |
|--|---------------------|
| Module Diagnostic | FD_SOURCE |
| Module Pronostic | FD_SOURCE |
| SCADA | FD_SOURCE |
| Module Gestion électronique de documents | DOCUMENT_RESSOURCE |
| Humain | PERSONNEL_RESSOURCE |
| Équipement | PHYSICAL_PRODUCT |
| Objet | Classe ontologique |
| GMAO | DOCUMENT_RESSOURCE |
| Base de données | DOCUMENT_RESSOURCE |
| Pièce de rechange | MATERIAL_RESOURCE |

TABLE 2 – Correspondances des modèles

entre ces deux unités se faisant alors au niveau 1.

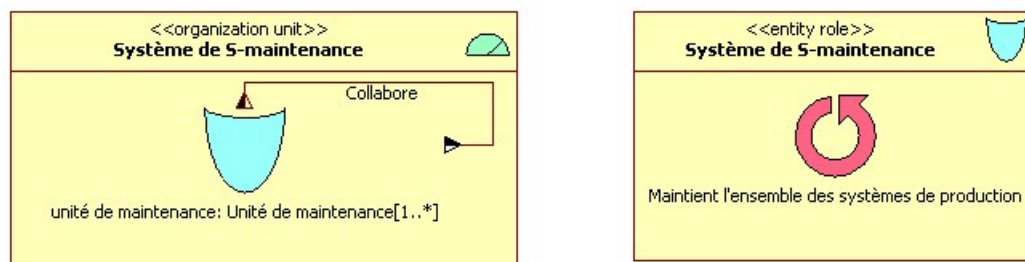


FIGURE 21 – Structure sociale niveau 2

11.3 Relation entre l'ontologie et la structure du modèle

Nous présentons dans le tableau 2 les correspondances entre les agents et objets du modèle de structure et les classes de l'ontologie (cf. 9.1 page 33).

11.4 Conclusion

Nous avons vu dans cette section la façon dont nous avons modélisé la plate-forme de s-maintenance avec les outils que nous avons choisis et qui sont décrits dans les sections précédentes. Nous avons développé le modèle de

la plate-forme telle qu'elle est décrite dans la partie III et avons su accorder notre modèle avec la représentation qu'en auront les agents.

12 Synthèse

Le premier travail effectué pour la contribution théorique au projet SMAC fut de choisir le paradigme de programmation qui allait nous servir dans nos travaux. Nous nous sommes naturellement tourné vers le multi-agents pour sa nature distribuée qui correspondait bien à la nature de la plate-forme à modéliser. Le projet SMAC étant un projet où la sémantique est primordiale, il nous a fallu étudier l'ontologie du projet et trouver une manière de l'intégrer dans notre simulateur multi-agents, ce que nous avons fait grâce aux spécifications de la FIPA. Nous avons ensuite cherché un moyen efficace et structuré de modéliser notre simulateur multi-agents. C'est dans les travaux de [Brocard, 2010], en partie basés sur le langage AML, que nous avons obtenu une méthode de modélisation à l'expressivité nécessaire pour caractériser la plate-forme SMAC. Enfin, nous avons modélisé la plate-forme grâce à cette méthode et avons lié ses concepts à l'ontologie du projet.

La suite de ce travail est l'implémentation de la plate-forme dans un framework multi-agents en suivant le modèle développé.

Cinquième partie

Implémentation

L'implémentation de la plate-forme est l'étape finale de notre travail. Nous n'avons pu la mener à son terme. Dans une première section nous justifions le choix de la plate-forme de développement. Nous verrons que ce choix a impliqué le développement préliminaire d'une implémentation du modèle AGR pour la plate-forme choisie. Ensuite, nous présenterons les difficultés rencontrées et les raisons de l'arrêt de l'implémentation dans son état actuel. Finalement, nous concluons cette partie par sa synthèse.

13 Choix de la plate-forme multi-agents

Nous avons mené des recherches sur les plates-formes de développement multi-agents afin de choisir laquelle serait la plus adaptée à nos besoins. Deux plates-formes ont particulièrement retenue notre attention : MadKit et JADE. Dans un premier temps nous présenterons MadKit, puis dans un deuxième temps JADE ; enfin, nous donnerons les raisons qui nous ont permis de choisir l'un plutôt que l'autre.

13.1 MadKit

MadKit (Multi-Agent Development Kit) est une plate-forme de développement de systèmes multi-agents basée sur le langage Java. Elle a été développée autour d'un modèle organisationnel, l'AGR (Agent, Groupe, Rôle). MadKit est constitué de plusieurs "couches" indépendantes visibles sur la figure 22. La base de la plate-forme est un micro-noyau permettant l'exécution des agents. Ce noyau est indépendant du reste de la plate-forme ce qui permet de le lancer seul si besoin est. En effet, le noyau gère les groupes et rôles locaux, le cycle de vie des agents et le passage de messages locaux. Les services plus complexes tel que le passage de messages distribués, la migration ou la surveillance d'agents sont fournis par des agents spécialisés à un méta-niveau [Gutknecht et al., 2000]. Enfin, MadKit propose une interface graphique et un certain nombre d'outils pour aider à la conception et l'implémentation des agents.

La principale raison pour laquelle nous nous sommes intéressés à MadKit est l'utilisation native de l'AGR dans la plate-forme. En effet, nous utilisons ce modèle organisationnel dans la conceptualisation et la modélisation de notre simulateur 10.2. Du point de vue de la programmation, les groupes

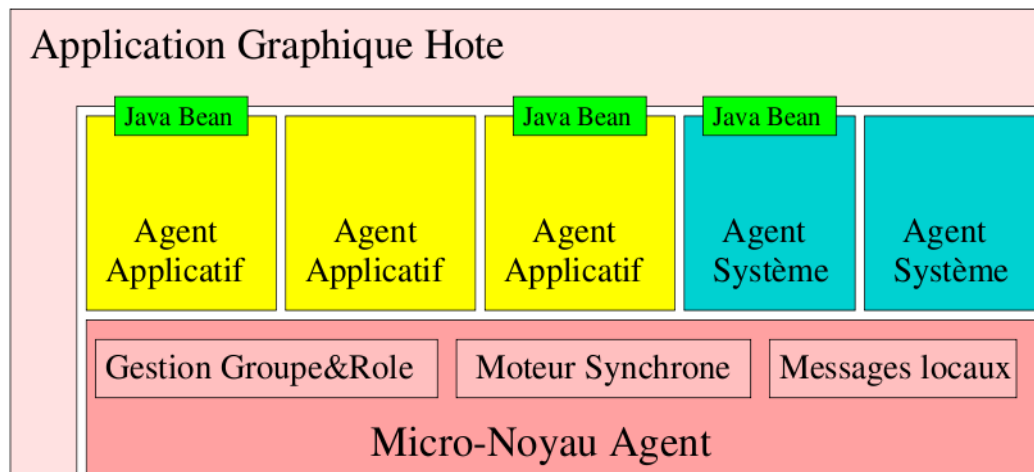


FIGURE 22 – Architecture générale de MadKit [Gutknecht et al., 2000]

et rôles sont créés à partir de chaînes de caractères. Lorsqu’un agent veut entrer dans un groupe il exécute la méthode “createGroup(..., String groupName, ...)” si le groupe existe déjà il le rejoint simplement, de même pour les rôles avec la méthode “requestRole(String groupName, String roleName, ...)”. Nous avons été séduits par cette façon simple de programmer la gestion des groupes et rôles. Néanmoins, un des axes particuliers de MadKit est de ne pas se conformer aux recommandations de la FIPA [Gutknecht et al., 2000], ce qui aurait pour conséquence, si nous choisissons cette plate-forme, le développement d’une bonne partie des structures et des mécanismes de contrôle recommandés par la FIPA afin de pouvoir utiliser correctement l’ontologie de SMAC.

13.2 JADE

JADE (Java Agent DEvelopment Framework) est une autre plate-forme de développement multi-agents. Un des principaux arguments défendu par ses auteurs pour son utilisation est qu’elle est conforme aux spécifications de la FIPA (cf.9.2 page 36). JADE propose une plate-forme distribuée exécutable sur plusieurs machines simultanément comme le montre la figure 23.

L’architecture interne(cf. fig24 de JADE comprend deux agents spéciaux : l’Agent Management Sytem (AMS) et le Directory Facilitator (DF). L’AMS est responsable de la création et de la gestion de nouveaux agents. Le DF est responsable de la gestion d’un annuaire de services proposés par ces agents. Tous les agents de la plate-forme distribuée peuvent communiquer entre eux

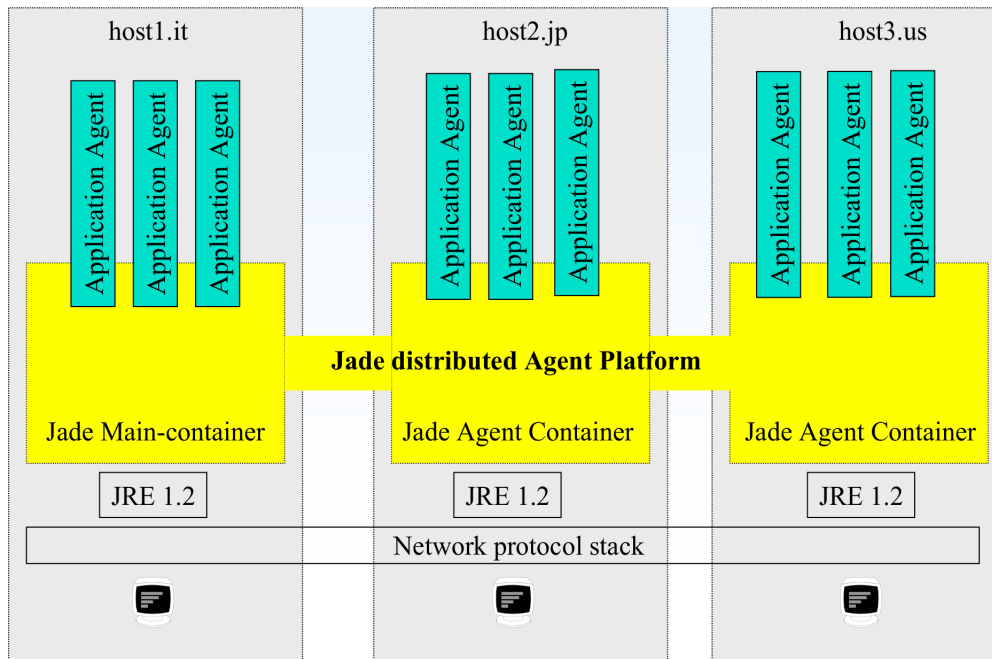


FIGURE 23 – Architecture de JADE [Bellifemine et al., 2001]

quelque soit la machine sur laquelle ils sont physiquement, grâce à une couche d'abstraction : l'Agent Communication Channel.

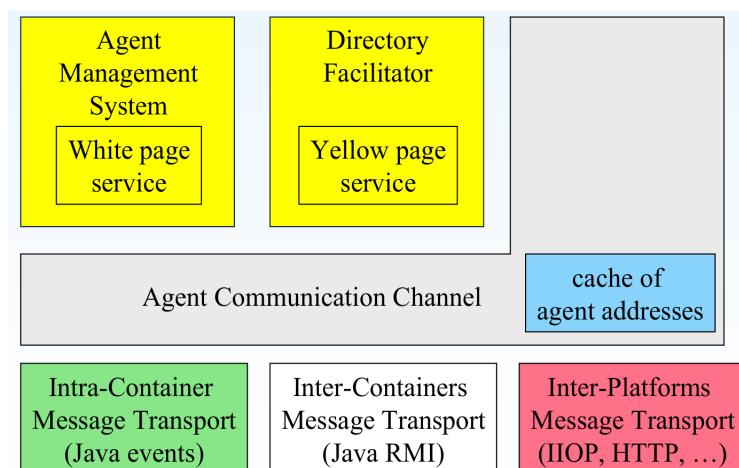


FIGURE 24 – Architecture interne de JADE [Bellifemine et al., 2002]

JADE permet d'utiliser nativement des ontologies. Un système de classes

de concepts et de prédicats a été développé afin de pouvoir traduire une ontologie sous forme de classe Java. Il est ainsi aisé d’y faire référence dans les classes implémentant les agents du système.

Les ontologies doivent absolument comporter des prédicats formulés selon une certaine forme c’est à dire contenus dans une classe implémentant l’interface “Predicate”. Ces prédicats sont, avec les “AgentAction”, indispensables aux actes de langage dans JADE. En effet, les méthodes de formatage du contenu des messages ne prennent comme argument que des instances de classe implémentant les interfaces héritées de “ContentElement” dont font partie les classes “Predicate” et “AgentAction”.

Exemple : considérons une classe (classe au sens de l’ontologie et pas une classe Java) “voiture” étant une sous-classe de la classe “véhicule”. Conceptuellement et même une fois transformée en classe Java une voiture hérite bien de la classe véhicule. Néanmoins, pour que JADE “comprenne” ceci, il est nécessaire de créer une classe implémentant l’interface “Predicate” qui dise explicitement que les deux classes ont un lien d’héritage.

Une fois l’ontologie créée correctement pour JADE, il est possible de formater les messages à partir de cette ontologie dans n’importe quel langage dont on a les codecs. Ces codecs se présentent sous la forme de greffons.

De nombreux greffons sont disponibles pour JADE.

13.2.1 Les greffons de JADE

La plate-forme JADE permet à de tierces parties d’ajouter des greffons (plugins) à celle-ci afin d’étendre ses capacités. Nous allons présenter ceux qui nous paraissent intéressants pour le projet SMAC.

Le plugin RDF

Le plugin RDF est un plugin de type codec pour JADE. Il permet d’encoder et de décoder des messages au format rdf. Pour SMAC, l’utilisation de ce plugin garantit l’interopérabilité sémantique. En effet, comme nous l’avons expliqué dans la section 2.1 RDF (utilisé avec XML) est le support du langage OWL. Il permet d’échanger des informations de type conceptuel entre les agents logiciels. Les agents ayant donc un langage commun et pouvant comprendre les mêmes concepts, l’interopérabilité sémantique est donc assurée.

L’Ontology Bean Generator

L’Ontology Bean Generator n’est pas à proprement parlé un plugin de JADE. C’est en fait un plugin du logiciel “Protégé”. “Protégé” est un logiciel

développé par l'université de Stanford ; il permet de créer de façon simple et graphique des ontologies. L'Ontology Bean Generator permet de transformer une ontologie écrite dans le langage OWL en une collection de classes Java représentant cette ontologie et normalement compatible avec JADE. L'ontologie de SMAC étant rédigée avec le langage OWL nous en sommes naturellement venus à nous intéresser à ce générateur.

Le plugin Sécurité

Les données du projet SMAC sont des données sensibles. Un équipementier ne tient pas à faire circuler des informations sur ses produits sur le réseau sans cryptage par exemple. Le plugin JADE-S permet ainsi d'apporter des notions de sécurité à JADE. Il met en place des processus d'authentification, de gestion des permissions et permet de crypter les messages en utilisant SSL.

13.3 Choix

Nous avons vu les atouts de deux plates-formes de développement de système multi-agents que sont JADE et MadKit. L'intérêt principal de MadKit tient dans le fait qu'il permet de gérer efficacement les groupes et les rôles. La plate-forme JADE quant à elle, permet, selon les normes de la FIPA, la gestion des ontologies, du langage RDF et de la sécurité.

Il nous est apparu que le meilleur choix qui s'offrait à nous pour l'implémentation du simulateur de SMAC était la plate-forme JADE car elle représentait un coup moindre en efforts de développement, MadKit manquant de trop de composantes sémantiques. N'ayant pas trouvé de travaux sur l'implémentation de l'AGR dans JADE nous avons réalisé cette implémentation nous mêmes. Celle-ci est décrite dans la section suivante.

14 Implémentation de l'AGR dans JADE

Pour les besoins de notre simulateur, nous avons développé une implémentation du modèle organisationnel AGR pour le système multi-agents JADE. Cette implémentation prend la forme de plusieurs dizaines de méthodes regroupées dans les classes portant les concepts de l'AGR : "AgentAGR", "GroupeAGR", "RoleAGR". Une quatrième classe a été développée à cause des spécificités de JADE, la classe "GroupAGRManager". Ces classes sont regroupées dans le package "agr".

D'un point de vue pratique, nous avons également développé un sous-package "exemple" qui contient un agent "prêt à programmer" : la classe "ReadyToProgramAgentAGR". Enfin, nous avons développé un sous-package

du package “example”, le package SW donnant un exemple simple mais assez complet au travers d’une communication entre sept agents.

Nous présenterons dans les trois premières sections les classes implémentées. Nous détaillerons ensuite les points importants de l’exemple SW, puis nous conclurons.

14.1 GroupAGRManager

La première classe développée est la classe “GroupAGRManager”. Cette classe gère et enregistre tous les groupes créés ainsi que tous les agents appartenant à un groupe. Pour que chaque agent puisse avoir accès à cette ressource commune de façon immédiate il était difficilement envisageable de procéder par envoi et réception de messages. Nous avons donc utilisé le design pattern du “singleton”. Ce design pattern permet, en invoquant le constructeur d’une classe, de référencer toujours la même instance de la classe, cette instance n’étant créée qu’à la première invocation du constructeur. Tous les agents ont donc accès à la même instance du manager de groupe. Ils peuvent donc invoquer ses méthodes publiques directement sans passer par des actes de langage qui seraient ici superflus.

La plupart des méthodes du manager sont soit des getters soit des méthodes d’affichage ou de test (avec un résultat booléen). Ces méthodes sont généralement utiles pour les agents qui gèrent les groupes dans le système.

Certaines méthodes tel que “joinGroup” n’ont normalement pas à être utilisées en dehors des appels à l’intérieur des méthodes de la classe “AgentAGR”. En effet, toutes les classes ont des références croisées les unes avec les autres. Par exemple le manager de groupe contient l’ensemble des références des groupes et des agents. Les groupes eux, contiennent l’ensemble des références des rôles que leurs agents jouent ainsi que les références des dits agents. Enfin, les agents contiennent l’ensemble des références des groupes auxquels ils appartiennent et des rôles qu’ils jouent.

Si “joinGroup” devait être utilisée pour forcer un agent à intégrer un groupe, l’état du système serait incohérent car l’agent ne saurait pas qu’il fait partie de ce groupe.

14.2 GroupAGR et RoleAGR

Les groupes comportent les méthodes nécessaires à leur gestion par les agents, ainsi qu’à la gestion de l’ensemble de leurs rôles. La plupart de ces méthodes sont utilisées par les méthodes de la classe “AgentAGR”. Il est

à noter que les agents référencés dans un groupe le sont par leur “AID” (AgentIdentifier). Cela s’explique par le fait que dans leurs communications les agents utilisent l’AID pour connaître l’expéditeur et le destinataire des messages.

Dans la gestion des rôles, on peut voir chaque groupe comme le manager des rôles qui lui sont locaux (nous rappelons qu’en AGR les rôles sont toujours locaux à des groupes) de la même manière que le “GroupAGRManager” gère les groupes.

14.3 AgentAGR

La classe “AgentAGR” contient toutes les méthodes nécessaires à la gestion des groupes, des rôles et des communications par les agents. Les agents peuvent ainsi créer, joindre, quitter ou détruire des groupes et des rôles. Ces méthodes ont été inspirées par les méthodes de MadKit. Comme elles, elles utilisent le nom des groupes et rôles pour leurs arguments, ce qui permet une programmation facile et rapide lors de leur utilisation dans des classes héritant d’AgentAGR.

Deux méthodes sont très importantes dans cette classe, ce sont les méthodes d’envoi de message : “sendAGROpt” et “sendAGRPes”. Le modèle AGR permet à un agent de communiquer seulement avec les agents partageant un groupe avec lui. Ainsi, la méthode d’envoi optimiste “sendAGROpt” provoque une exception si la règle n’est pas respectée alors que la méthode pessimiste ignore les destinataires incorrects pour ne faire parvenir les messages qu’à ceux autorisés à les recevoir.

Pour éviter au programmeur de fastidieuses lignes de code, nous avons également implémenté des méthodes pour la diffusion de messages à tous les agents d’un groupe ou encore tous les agents jouant le même rôle.

14.4 Le package SW

Le package SW présente un dialogue entre deux agents. A la fin de ce dialogue un agent diffuse une information à son groupe. En réponse à cette information les membres quittent le groupe.

Outre la démonstration concernant les envois et réceptions de messages ainsi que les arrivées ou sorties de groupe dans les classes dérivées d’AgentAGR, c’est la classe SWLoader qui est intéressante ici.

L’instance de cette classe est un Agent pas un AgentAGR. Elle sert à instancier et charger les agents dans le bon ordre. En effet, si l’on ne met pas de verrous durant la phase de démarrage de chaque agent on se heurte à des pro-

blèmes de synchronisation. Typiquement, un agent souhaitera communiquer avec un autre qui n'est pas encore arrivé dans son groupe.

14.5 Conclusion

Nous avons implémenté le modèle AGR dans la plate-forme JADE en nous inspirant de ce qui avait été fait avec MadKit. Néanmoins, nous n'avons pas dupliqué exactement le système de gestion de groupe de MadKit. Nous avons au contraire laissé volontairement beaucoup de possibilités aux développeurs qui souhaitent utiliser cette implémentation et l'avons dotée à cet effet de la meilleure documentation pour qu'il soit aisé de la comprendre et au besoin de la modifier. C'est sur la base de cette implémentation du modèle AGR que nous avons pris appui pour tenter d'implémenter la plate-forme de SMAC.

15 Implémentation de notre modèle et difficultés rencontrées

Nous n'avons pas réussi à mener à bien l'implémentation du simulateur de SMAC. Nous allons exposer les difficultés qui nous ont bloqués dans l'avancement de nos travaux et présenter le peu de travail effectué sur cette partie, puis nous concluons

15.1 Les problèmes avec l'ontologie et l'Ontology Bean Generator

Comme nous l'avons décrit à la section 13.2 pour pouvoir être utilisée dans les actes de langage l'ontologie doit comporter des prédicats bien formés. Du point de vue de JADE, ce sont des classes implémentant l'interface "Predicate", du point de vue de l'ontologie les prédicats sont des sous classes d'une classe "predicate". Les ontologies écrites en OWL ne sont pas naturellement modélisées de la sorte et celles de SMAC ne faisaient pas exception. Les prédicats sont des propriétés de classe.

De plus, il est impossible de transformer ces propriétés en classes (pour en faire des sous classe de "predicate") sans perdre le lien qu'elles représentaient entre les classes dont elles étaient les propriétés. Nous avons donc essayé d'ajouter des sous classes de "predicate" au fur et à mesure de notre implémentation. Cette initiative n'est pas allée très loin puisque nous avons été arrêtés par des exceptions provoquées par JADE. Ces exceptions concernent la classe "Schema". Dans une ontologie sous forme de classe Java pour JADE

chaque concept est associé à un schéma. Nous n'avons pas réussi à découvrir comment régler ce problème.

Nous suspectons l'Ontology Bean Generator de générer des classes qui ne sont pas totalement compatibles avec JADE. Le générateur n'a pas été mis à jour depuis des années, de plus, la documentation qui y est associée est très pauvre. Nous avons, en dernier recours, chercher les sources du générateur, mais sans succès.

15.2 Les éléments partiellement implémentés

Voici les quelques éléments de la plate-forme partiellement implémentés

15.2.1 L'OntologyAgent

Comme préconisé par la spécification FIPA-OSS, nous avons cherché à mettre en place un OA avec lequel les autres agents de la plate-forme auraient pu interagir afin de demander et de fournir des informations relatives à l'ontologie. N'ayant pas pu utiliser cette ontologie, cet agent n'a pas pu être terminé.

15.2.2 L'agent Humain

Nous avons tenté de développer l'agent "Humain" en même temps que l'OA afin de pouvoir les faire interagir. Nous avons prévu de reprendre les travaux menés par D.Saint Voirin sur son simulateur multi-agents dont nous avons les sources, ceci afin d'implémenter le niveau de connaissance des agents humains.

15.2.3 La classe capteur

Nous avons créé une classe capteur pour simuler les changements des conditions des équipements. Chaque capteur est un thread qui met à jour une valeur à intervalle fixe défini lors de la construction de celui-ci. Nous avons également créé deux classes filles "CapteurHumidite" et "CapteurTemperature" avec des valeurs par défaut (par exemple l'humidité a une valeur maximale de 100%).

15.2.4 Les classes SCADA et MonitorSCADA

La classe Monitor est composée de deux seuils : un seuil mini et un seuil maxi. Elle comprend également un écouteur sur un capteur. A chaque cap-

teur est attribué un monitor.

Un SCADA est un agent qui est composé d'une collection de monitors. Lorsqu'un seuil est dépassé, le monitor fait envoyer par le SCADA, auquel il appartient, une alerte. L'ontologie de SMAC n'ayant pas pu être intégrée dans le simulateur, le corps du message n'a pas pu être créé.

15.3 Conclusion

Pour rendre possible l'implémentation de cette plate-forme, nous pensons qu'un travail en profondeur sur l'ontologie de SMAC et sur la forme qu'elle devra prendre pour être utilisable avec JADE doit être mené. Même si nous n'avons pas eu l'occasion d'utiliser les groupes et les rôles dans ce développement, nous avons foi en notre travail sur l'implémentation du modèle AGR pour JADE et le recommandons pour l'implémentation de la plate-forme SMAC.

16 Synthèse

Le premier travail après avoir pris la décision d'implémenter notre modèle a été de choisir une plate-forme de développement propice à accueillir le simulateur de SMAC. Notre choix s'est porté sur JADE pour ses qualités dans la gestion de la sémantique et de son respect des normes de la FIPA. Notre modèle théorique se basant sur le modèle AGR, nous avons développé des classes Java permettant la gestion efficace des groupes et des rôles au sein de JADE. Pour finir, nous avons commencé l'implémentation de la plate-forme mais avons été bloqués par des problèmes aussi bien techniques que conceptuels.

Sixième partie

Conclusion

Ce stage a débuté par une recherche bibliographique sur les différents aspects du projet SMAC. Nous avons ainsi cherché à comprendre le projet et à dégager des pistes de réflexion pour mener à bien notre travail. Nous avons donc étudié les principes de la e-maintenance et de la s-maintenance. Nous avons ainsi intégré les principes d'interopérabilité sémantique et de collaboration. Nous avons approfondi ces deux notions en étudiant d'une part les travaux menés dans le web sémantique, notamment sur les langages ontologiques, et d'autre part sur les systèmes de maintenance coopératifs et collaboratifs présents dans la littérature. Cette étude nous a amené à considérer les travaux de D.Saint Voirin et d'A.Seguy comme des pistes possibles, dont nous nous sommes d'ailleurs inspiré, pour nos travaux.

Nous avons dans un deuxième temps cherché à mieux comprendre le projet SMAC plus en détail. Nous avons ainsi tissé les liens entre le projet réel et les concepts découverts lors de notre recherche bibliographique.

Fort de cette compréhension, nous nous sommes attelé à produire un modèle aussi proche que possible du système réel en vue de l'implémentation d'une simulation de cette plate-forme. Pour se faire nous avons tout d'abord pris la décision d'utiliser le paradigme multi-agents pour la modélisation de ce système fortement distribué. Une fois ce choix arrêté et après avoir approfondi nos connaissances dans la échanges sémantiques inter-agent, nous nous sommes rendu compte que le système SMAC correspondait à la définition d'un système complexe, ce qui nous a amené à nous intéresser aux travaux de T.Brocard sur la modélisation de systèmes complexes par l'utilisation du paradigme multi-agents et du langage AML. Avec alors, toutes les cartes en mains, nous avons réalisé la modélisation de la plate-forme SMAC.

Cette modélisation devait être implémentée dans un simulateur. Nous avons commencé par chercher une plate-forme de développement multi-agents. Notre choix s'est porté sur la plate-forme JADE car elle disposait de beaucoup d'avantages du point de vue de la gestion des échanges sémantiques entre agents. Il manquait néanmoins un élément important dans cette plate-forme par rapport à notre modèle : la gestion des groupes et des rôles définis dans le modèle AGR et dont nous nous sommes servi dans notre modélisation. L'implémentation de l'AGR dans JADE a constitué un travail préparatoire

à l'implémentation de la simulation de SMAC.

Nous n'avons pas réussi à mener à bien cette implémentation. Nous avons rencontré beaucoup de problèmes techniques qui ont sensiblement ralenti notre progression dans l'implémentation de cette simulation jusqu'à ce que le temps nous manque pour pouvoir l'achever. Nous croyons néanmoins que le travail de modélisation ainsi que l'AGR pour JADE permettront d'implémenter cette simulation ou d'autres concernant les systèmes complexes.

Il y a plusieurs voies à explorer dans la continuité de ce projet. La première est bien entendu de finaliser la simulation de la plate-forme. Une autre voie peut également être l'affinage de la modélisation de cette plate-forme. En effet notre modélisation ne comporte que trois niveaux. Il est possible, en spécifiant plus les agents modélisés au niveau 0, d'améliorer la modélisation et peut-être de découper le système en un, voir en deux niveaux d'abstraction supplémentaires.

Dans la perspective où la simulation révélerait un réel intérêt pour l'interopérabilité sémantique entre les agents, on pourrait envisager d'agentifier le système cible et ainsi de mettre des agents au cœur de la plate-forme SMAC.

Enfin, un travail de fond qui pourrait être réalisé, serait d'harmoniser la façon de créer des ontologies et celle de les utiliser dans des système multi-agents tel que JADE.

Nous voudrions terminer cette conclusion en exposant les apports personnels que ce stage a eu pour nous.

Ce stage nous a fait abordé quelques aspects de la recherche en informatique. Nous avons découvert les travaux d'une équipe de chercheurs et leur façon de travailler. Nous avons également remarqué la facilité avec laquelle il était possible de s'égarer sur de mauvaises voies, étant nous mêmes ceux qui traçons les chemins. Et c'est d'ailleurs grâce à cela que nous avons trouvé un grand intérêt à ce stage.

Références

- [Afnor, 2001] Afnor, EN, N. (2001). 13306," terminologie de la maintenance"(ancienne norme nf x60-010), ed. *Afnor, Paris*.
- [Aristeidis, 2010] Aristeidis, M. (2010). Smac deliverable 3.1. *Inconnu*, inconnu :1–33.
- [Bellifemine et al., 2002] Bellifemine, F., Caire, G., Rimassa, G., Poggi, A., Trucco, T., Cortese, E., Quarta, F., Vitaglione, G., Lhuillier, N., and Picault, J. (2002). Java Agent Development Framework. *TILAB Italia*, <http://jade.cse.it>, status, 10 :2002.
- [Bellifemine et al., 2001] Bellifemine, F., Poggi, A., and Rimassa, G. (2001). Developing multi-agent systems with JADE. *Intelligent Agents VII Agent Theories Architectures and Languages*, pages 42–47.
- [Brocard, 2010] Brocard, T. (2010). Definition d’un modele agent de micro-systemes complexes pour leur verification, validation et simulation. inconnue.
- [Ferber, 1995] Ferber, J. (1995). *Les systemes multi-agents : vers une intelligence collective*. InterEditions Paris.
- [FIPA, 2001] FIPA (2001). Fipa ontology service specification.
- [FIPA, 2002] FIPA (2002). Fipa acl message structure specification.
- [Garcia et al., 2004] Garcia, E., Guyennet, H., Lapayre, J., and Zerhouni, N. (2004). A new industrial cooperative tele-maintenance platform. *Computers & Industrial Engineering*, 46(4) :851–864.
- [Gutknecht et al., 2000] Gutknecht, O., Ferber, J., and Michel, F. (2000). MadKit : une architecture de plate-forme multiagent générique. *Rapport de recherche*, 61.
- [Halevy et al., 2005] Halevy, A., Ives, Z., Suciu, D., and Tatarinov, I. (2005). Schema mediation for large-scale semantic data sharing. *The VLDB Journal*, 14(1) :68–83.
- [Koc et al., 2003] Koc, M., Ni, J., Lee, J., and Bandyopadhyay, P. (2003). Introduction of e-manufacturing. In *Proceedings of the International Conference on Frontiers on Design and Manufacturing*. Citeseer.
- [Lai and Chu, 2000] Lai, H. and Chu, T. (2000). Knowledge management : a review of theoretical frameworks and industrial cases. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, page 10.
- [Larousse, 2010] Larousse (2010). *Larousse 2010*. Larousse.

- [Lee, 2003] Lee, J. (2003). E-manufacturing-fundamental, tools, and transformation. *Robotics and Computer Integrated Manufacturing*, 19(6) :501–507.
- [Lee J, 2008] Lee J, H. W. (2008). *Complex System Maintenance Handbook*, chapter New Technologies for Maintenance, pages 49–78. Springer.
- [Maedche and Staab, 2000] Maedche, A. and Staab, S. (2000). Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA*.
- [Marquez, 2007] Marquez, A. C. (2007). *The Maintenance Management Framework*, chapter The E-maintenance Revolution, pages 305–327. Springer.
- [Maurizio Tomasella, 2006] Maurizio Tomasella, Jacopo Cassina, A. M. M. M. (2006). Dr9.2 : Specification of the system object model.
- [Muller, 2005] Muller, A. (2005). *Contribution à la maintenance prévisionnelle des systèmes de production par la formalisation d un processus de pronostic*. PhD thesis, Université Henri Poincaré, Nancy.
- [Rasovska, 2006] Rasovska, I. (2006). *Contribution à une méthodologie de capitalisation des connaissances basée sur le raisonnement à partir de cas : Application au diagnostic dans une plateforme d e-maintenance*. PhD thesis, UFR des Sciences et Techniques de l Université de Franche-Comté.
- [Rasovska et al., 2007] Rasovska, I., Chebel Morello, B., and Zerhouni, N. (2007). Classification des différentes architectures en maintenance. In *Actes du 7ème Congrès International de Génie Industriel, GI'2007. 7ème Congrès International de Génie Industriel, GI'2007, Trois Rivières.*, pages sur CD ROM – 12 pages, Québec Canada. UQTR.
- [Saint-Voirin, 2006] Saint-Voirin, D. (2006). *Contribution a la modelisation et a l analyse des systemes cooperatif : application a la e-maintenance*. PhD thesis, Université de Franche-Comté, Besançon.
- [Seguy, 2008] Seguy, A. (2008). *Decision collaborative dans les systemes distribues - Application a la e-maintenance*. PhD thesis, Université de Toulouse.
- [Simon, 1996] Simon (1996). Knowledge acquisition and modeling for corporate memory : lessons learnt from experience. *Proc. Of KAW 96, Banff, Canada*, 1 :11–18.
- [Trappey et al., 2008] Trappey, A., Hsiao, D., Ma, L., Chung, Y., and Kuo, Y. (2008). *Global Perspective for Competitive Enterprise, Economy and Ecology*, chapter Agent-based Collaborative Maintenance Chain for Engineering Asset Management, pages 29–41. Springer.

- [Ucar and Qiu, 2005] Ucar, M. and Qiu, R. (2005). E-maintenance in support of E-automated Manufacturing Systems. *Journal of the Chinese Institute of Industrial Engineers*, 22(1) :1–10.
- [Whitestein, 2004] Whitestein (2004). Aml – the agent modeling language.
- [Wikipédia, 2010] Wikipédia (2010). Wikipédia.
- [Yu et al., 2003] Yu, R., Iung, B., and Panetto, H. (2003). A multi-agents based e-maintenance system with case-based reasoning decision support. *Engineering Applications of Artificial Intelligence*, 16(4) :321–333.

Résumé

Ce rapport décrit le travail réalisé pour le stage d'initiation à la recherche effectué au LIFC à la fin de notre deuxième année de master informatique. Ce stage avait pour but d'éprouver l'interopérabilité sémantique ainsi que les processus collaboratifs au sein d'une plate-forme de s-maintenance par la réalisation d'une simulation.

Pour répondre à cette problématique, nous avons effectué un travail de recherche bibliographique sur les sujets de la e-maintenance, la collaboration et la sémantique dans les systèmes informatiques. Nous avons fait le choix de modéliser la plate-forme en utilisant le paradigme multi-agents. Nos travaux nous ont menés à considérer les outils de modélisation de systèmes complexes comme étant les mieux adaptés à notre situation.

Une fois notre modélisation achevée nous avons tenté d'implémenter la simulation. Ce travail nous a amenés à développer des classes de gestion du modèle AGR dans le système multi-agents JADE. Nous n'avons pas pu terminer l'implémentation de cette simulation pour des raisons techniques, mais avons donné le matériel nécessaire pour la réaliser.

Mots clés : s-maintenance, e-maintenance, collaboration, interopérabilité sémantique , système multi-agents, système complexe

Abstract

This report describe the work done during the internship in the LIFC laboratory for our final year of master degree. The aim of the work executed was to test the pertinence of the semantic interoperability and of collaborative process in a s-maintenance platform using a simulation.

We performed a bibliographic work on e-maintenance, collaboration and semantic in order to address this issue. We chose to design the platform using the multi-agent paradigm. We noted that the use of complex system model was relevant for our model.

Then we try to implement the simulation, but we failed because of technical issues. Nevertheless, we developed management classes of the AGR model in the MAS JADE to support this work.

Keywords : s-maintenance, e-maintenance, collaboration, semantic interoperability, multi-agent system, complex system