RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université LARBI BEN M'HIDI

-Oum El Bouaghi-

Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie

Département de Mathématiques et Informatique

N° d'ordre **:** .....

Série**:** .....

Contrôlabilité des Systèmes Multi-Agents Ouverts

THÈSE **présentée par :**

CHEBOUT Mohamed Sedik

Pour l'obtention du diplôme de

DOCTORAT EN SCIENCE en informatique

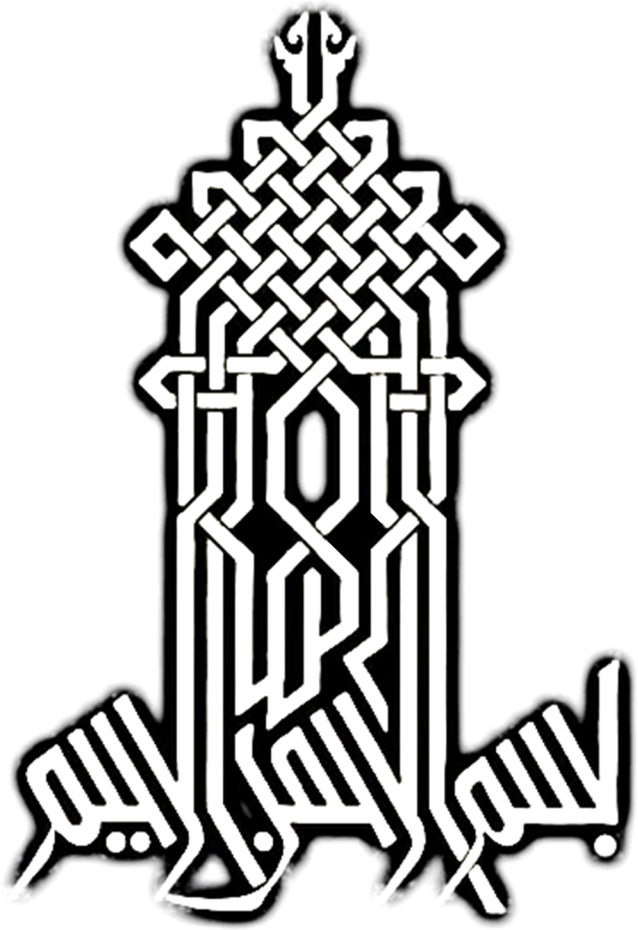
(Option : **Intelligence Artificielle et Imagerie**)

Soutenue publiquement le : **16/12/2019**.

Devant le jury composé de :

Président	**DERDOURI Lakhdar**	***Professeur***	*Université d'Oum El Bouaghi*
Directeur de thèse	**MOKHATI Farid**	***Professeur***	*Université d'Oum El Bouaghi*
Co-directeur de thèse	**BABAHENINI Mohamed Chaouki**	***Professeur***	*Université de Biskra*
Examinateurs	**BENNOUI Hammadi**	***Professeur***	*Université de Biskra*
	MERAH ElKamel	*MCA*	*Université de Khenchela*
	GUERRAM Tahar	*MCA*	*Université d'Oum El Bouaghi*
Invité	**BADRI Mourad**	***Professeur***	*Université du Québec à Trois-Rivières, Qc, CANADA*

2019-2020


بسم الله الرحمن الرحيم

Remerciements

De prime abord je tiens à exprimer ma profonde reconnaissance à mon encadrant, professeur **MOKHATI Farid**, *d'avoir accepté de diriger mes travaux de thèse tout au long de mon parcours doctoral. En effet, j'étais toujours sous le parrainage du Pr.* **MOKHATI** *depuis mon ingéniorat et mon magister. Une telle supervision m'a permis d'apprendre énormément de choses non seulement dans le cadre académique et professionnel, mais également dans la vie quotidienne.*

Mon cher professeur, je vous remercie démesurément.

Je voudrais également remercier mon co-encadrant de thèse le professeur : **BADRI Mourad** *de l'université du Québec à trois rivières, pour ses qualités humaines et scientifiques. En fait, une telle connaissance me permet, à titre personnel, de progresser scientifiquement dans le bon sens et de bien pratiquer comment faire les choses de la manière la plus appropriée.*

Je tiens également à remercier grandement le professeur **BABAHENINI Mohamed Chaouki**, *mon co-encadrant de thèse, qui m'a accompagné chaleureusement depuis ma première année d'inscription à l'université de BISKRA. J'apprécie son soutien et son engagement professionnel. Nos liens sont désormais indéfectibles.*

Mes sincères remerciements sont adressés aux membres de jury qui m'ont fait l'honneur d'avoir accepté d'évaluer mes travaux de thèse :

Pr. **DERDOURI Lakhdar**
Pr. **BENNOUI Hammadi**
Pr. **MERAH ElKamel**
Pr. **GUERRAM TAHAR**

Leurs conseils et suggestions contribueront, très certainement, à l'enrichissement et à l'amélioration de ce travail.

Un remerciement très particulier est dédié à Dr. **MARIR Toufik**, *mon enseignant et mon collègue à l'université d'Oum El Bouaghi. J'ai apprécié l'atmosphère scientifique qu'il m'a offerte et le temps qu'il m'a consacré au sein du laboratoire ReLa(CS)2.*

Toute ma reconnaissance à ma femme qui n'a cessé de s'inquiéter pour mes études.

Finalement, un remerciement chaleureux à ma famille pour leur soutien inconditionnel tout au long de cette période de rédaction.

Dédicaces

À mes parents.

À mes puces : Lina El-Batoul et Sajed Mohamed Ikbal.

À ma femme Noussaiba.

À mes beaux-parents

Ainsi qu'à tous ceux qui m'aiment.

ملخص

الأعمال البحثية المقدمة في هذه الأطروحة تندرج في إطار هندسة البرمجيات الموجهة للوكلاء. بالتحديد ركزنا على إشكالية التحكم في الأنظمة متعدة الوكلاء المفتوحة. تعتبرالسيطرة على سلوكات الأنظمة المتعددة الأعوان المفتوحة عملية بالغة الصعوبة بالنظر إلى الخصائص الجوهرية التي يتمز بها هذا النوع من الأنظمة كـ : الإنفتاح، الديناميكية، عدم الحتمية، عدم التجانس وكذلك خاصية بروز تصرفات غير متوقعة وغير مصممة مسبقا. من أمثلة الأنظمة متعددة الأعوان المفتوحة نجد الأنظمة التي يتم برمجتها اعتمادا على أنموذج تنظيمي (بالتحديد الأنموذج الموسوم بـ : **AGR** عون، مجموعة، دور). بداية قمنا بتقسيم إشكالية التحكم إلى جزئين أساسين : المراقبة والتحكم. تقوم المراقبة على التحليل والدراسة الآنية لتصرف النظام سواء على مستوى النظام في عمومه أو على مستوى العون بالتحديد. المقاربة المقترحة لإشكالية المراقبة تم تجسيدها عن طريق أداة برمجية تحت اسم : **RT-MTOMAS**. بالمقابل مقاربة التحكم المقترحة تحمل اسم : **NorCtrl4OMAS** وقد قمنا بتجسيدها أيضا عن طريق أداة برمجية مختصة. تقوم مقاربة التحكم على استخدام المعايير كوسيلة تحكم أساسية. وكآخر اسهاماتنا في الموضوع، قمنا بتطبيق المقاربتين المقترحتين على مجموعة من دراسات الحالة على منصة **MaDKit** .

الكلمات المفتاحية:

AGR (عون، مجموعة، دور)، الأنظمة المتعددة الأعوان المفتوحة المعتمدة على **AGR**، المراقبة، المعايير، منصة **MaDKit**.

Résumé

Les travaux présentés dans cette thèse se placent dans le contexte du génie logiciel orienté-agent. Nous nous focalisons sur le problème de contrôlabilité des systèmes multi-agents ouverts (SMA ouvert). La maitrise du comportement des SMA ouverts est une tâche très complexe vis-à-vis les caractéristiques inhérentes à ce type des SMA telles que : l'ouverture, la dynamique, le non-déterminisme, l'hétérogénéité, l'émergence, etc. Un type particulier des SMA ouverts est celui implémenté en se basant sur un modèle organisationnel (bien particulièrement le modèle AGR : Agent, Group, Rôle). Nous partons par diviser le problème de contrôle des SMA ouverts basés-AGR en deux phases complémentaires : le monitoring et le contrôle proprement dit. Ainsi, nous avons proposé deux approches différentes pour chaque phase. Le monitoring permet d'analyser et d'examiner dynamiquement le comportement des SMA ouverts que ce soit pour le niveau système ou agent. L'approche de monitoring proposée a été concrétisée par un outil logiciel baptisé RT-MTOMAS. En revanche, l'approche de contrôle proposée a été déployée sous le nom NorCtrl4OMAS et est validée, également, avec un outil logiciel dédié. NorCtrl4OMAS utilise principalement les normes comme étant un mécanisme de contrôle. Nous avons étudié l'applicabilité des deux approches proposées sur des études de cas concrètes sous la plateforme MaDKit.

Mots-clés :

AGR (Agent, Group, Rôle), SMA Ouverts basés-AGR, Monitoring, Normes, MaDKit.

Abstract

Research accomplished in this thesis is placed in the context of agent oriented software engineering. We focus, mainly, on the controllability problem of Open Multi-Agent Systems (Open MAS). Mastering the behaviour of Open MAS is a hard task with respect to inherited characteristics like: openness, dynamic, non-determinism, heterogeneity, emergency, etc. A particular kind of Open MAS is the one implemented based on AGR (Agent, Group, Role) organizational model. We start by dividing control problem of AGR-based Open MAS into two complementary phases: monitoring and controlling. Also, we have proposed a different approach for each phase. Monitoring consists of analysing and reviewing dynamically the behaviour of Open MAS for both agent and system levels. Proposed monitoring approach is supported by a software tool baptized (RT-MTOMAS). However, proposed control approach has been deployed under the name: NorCtrl4OMAS and validated with a dedicated software tool. NorCtrl4OMAS uses norms as a control mechanism. We investigated the applicability of the two approaches using concrete case studies under MaDKit agent platform.

Key words :

AGR (Agent, Group, Role), AGR-based Open MAS, Monitoring, Norms, MaDKit.

Table des matières

Table des figures

Liste des tables

Liste des abréviations

Acronyme	Explication
AGR	Agent, Group, Role
AGRE	AGR + Environment
AGRS	AGR + Service
AMELI	Agent-based Middleware for ELectronic Institutions
AOCG	Agent Oriented Call Graph
AUML	Agent UML
BDI	Belief-Desire-Intention
BOID	Belief-Obligation-Desire-Intention
BNF	Backus-Naur Form
CFP	Chapitre 4 : Call For Proposal
	Chapitre 5 : Call For Paper
CRS	Conference Review System
DynaCrom	DYNAmic Contextual Regulation information provision in Open MASs
FIPA-ACL	Foundation for Intelligent Physical Agents-Agent Communication Language
IA	Intelligence Artificielle
IAD	IA Distribué
IE	Institution Electronique
IHM	Interface (ou Interaction) Homme-Machine
IOA	Instrumentation Orienté-Aspect
JADE	Java Agent DEvelopment framework
JESS	JAVA Expert System Shell
JVM	JAVA Virtual Machine
LHS	Left-Hand Side
LISP	LISt Processing
MaDKit	Multi-agent Development Kit
MaNEA	Magentix2 Norm-Enforcing Architecture
MI	Modèle Intéractionnel
MO	Modèle Organisationnel
MOA	Monitoring Orienté-Aspect
MS	Modèle Sociale
NBDI4JADE	Normative BDI for JADE
NorCtrl4OMAS	Norms-based ConTRoL for Open Mult-Agent Systems

NorJADE	**Normative JADE**
NormML	**Norm Modeling Language**
OP-RND	**Obligation-Prohibition Recommended Neutrality Disliked**
POA	**Programmation Oriénté-Aspect**
POO	**Programmation Orienté-Objet**
RDP	**Résolution Distribué des Problèmes**
RHS	**Rigth-Hand Side**
RT-MTOMAS	**RealTime Monitoring Tool for Open Multi-Agent Systems**
RT-MTOMAS DASD	**RT-MTOMAS Dynamic Agent Sequence Diagram**
SCAAR	**Self-Controlled Autonomous Agents geneRator**
SDL	**Standard Deontic Logic**
SMA	**Système Multi-Agents**
SMACA	**Systèmes Multi-Agents Centrés Agent**
SMACO	**Systèmes Multi-Agents Centrés Organisation**
SSM	**Système Sous Monitoring**
TSSCP	**Target System State Checking Process**
UML	**Unified Modeling Language**
VO	**Virtual Organisation**
XML	**eXtensible Markup Language**

INTRODUCTION GÉNÉRALE

1. Contexte général de la problématique

L'avènement de l'Intelligence Artificielle Distribuée (IAD) était, principalement, pour combler les lacunes de l'Intelligence Artificielle (IA) classique en proposant la distribution de la connaissance sur plusieurs entités. Les Systèmes Multi-Agents (SMA) représentent, quant à eux, l'un des trois principaux axes de l'IAD, à l'instar de la Résolution Distribuée des Problèmes (RDP) et de l'Intelligence Artificielle Parallèle (IAP). À cet effet, les recherches dans le domaine de l'IAD et les SMA se focalisent généralement sur la manière de répartir un problème sur un ensemble d'entités coopérantes afin d'assurer, pour un système donné, l'achèvement de son objectif (Gasser & Bond, 1988). Le but pour lequel un SMA est conçu sera, donc, assuré par la combinaison des comportements des agents qui le constituent. Cependant, le nombre d'entités qui composent un SMA est un paramètre qui n'est pas toujours connu à priori particulièrement dans le cas des SMA ouverts où la composition du système change dynamiquement.

Dans l'ingénierie des SMA, des méthodologies, des outils et des langages ont été proposés dans le but de développer des SMA de qualité. Ainsi, ces méthodologies proposent des techniques qui prennent en considération les spécificités des SMA telles que : la distribution des tâches, l'interaction entre les agents, l'environnement qui regroupe les agents et même l'ouverture qui caractérise ce type de systèmes. Cependant, les tâches de conception et de développement des SMA ouverts sont réalisées par deux équipes différentes (Gonzalez-Palacios & Luck, 2006). Autrement dit, les composants de tels systèmes dans la phase d'implémentation ne sont pas les mêmes que dans la phase de conception. Par ailleurs, le nombre d'agents d'un SMA ouvert change dynamiquement durant son exécution (García-Camino, Noriega, & Rodríguez-Aguilar, 2005).

Les agents dans un SMA ouvert sont, par définition, de différents types et architectures et disposent chacun de son propre but. C'est-à-dire, ils sont conçus par différentes équipes dont la mise en œuvre nécessite l'intervention des méthodes et technologies différentes (concepts, architectures, langages de programmation, etc.) (Artikis, 2012) (Artikis, Sergot, Pitt, Busquets, & Riveret, 2016), parfois une situation de conflits aura lieu si deux ou plusieurs agents veulent arriver à des buts contradictoires (Hewitt, 1991). Bien que l'aspect interne des agents soit inaccessible, la communication des agents est le seul moyen avec lequel on peut observer le comportement des agents à travers l'échange d'informations (Paes, et al., 2005). Ainsi, la manière d'interaction et de coordination des agents au sein d'un SMA ouvert doit être observée et contrôlée dans le but de faire face à la nature hétérogène de ses composants d'une part et d'essayer de comprendre les phénomènes émergents qui en découlent d'autre part. Par conséquent, l'ouverture sans contrôle peut conduire à des comportements chaotiques, tel que mentionné dans (Esteva, Rosell, Rodriguez-Aguilar, & Arcos, 2004).

Un autre point soulevé pendant l'étude des SMA ouverts c'est l'hyper difficulté (voire l'impossibilité des fois) de prévoir à un moment donné durant l'exécution du système son état prochain. En d'autres termes, la mise en interaction des agents qui ont été, à la base, conçus par différents concepteurs mène, certainement, à des situations d'anomalies et génère en conséquence des problèmes sérieux. Nous citons à titre d'exemple le problème de la sécurité du système provoqué par la liberté de communication où l'information est changée sans qu'elle soit contrôlée. C'est la raison pour laquelle un agent doit vérifier, initialement, la qualification de son interlocuteur (Ferber, Gutknecht, & Michel, 2004). Ainsi, l'absence d'une entité centrale (une autorité) qui est à la charge du contrôle global du système risque d'avoir des agents dans le système qui se comportent comme des pirates et agissent frauduleusement. Ainsi, le caractère ouvert des SMA offre la possibilité aux agents inconnus de perturber le système et de rediriger son comportement global vers des états imprévisibles et non conçus préalablement.

Pour surmonter ces difficultés, les chercheurs dans le domaine des SMA comme : (Ferber & Gutknecht, 1998), (Jennings, 2000), (Ferber, Gutknecht, & Michel, 2004), (Paes, et al., 2005), (Mansour & Ferber, 2007), (Fogués, Alberola, Such, & García-Fornes, 2010) et (Xu, Zhang, & Patel, 2007) ont pensé à doter les SMA ouverts d'un niveau organisationnel qui permet aux agents d'éviter les interactions négatives et favoriser, en revanche, les interactions utiles. Une telle mesure organisationnelle est dédiée bien particulièrement aux situations où les agents se coordonnent pour réaliser un but global commun. La modélisation organisationnelle des SMA ouverts représente une alternative intéressante pour combler le fossé entre l'ouverture et l'état global souhaité du système.

Dans la littérature, plusieurs définitions du terme « organisation » ont été proposées à l'instar de (Gasser , 1992) et (Wooldridge, Jennings, & Kinny, 2000). Ces travaux ont clairement exprimé l'organisation en termes de rôles et de buts attendus par ces rôles. Cependant, ils n'ont pas mentionné clairement comment ces rôles sont regroupés et assignés aux agents. De plus, aucun détail n'a été fourni sur la manière dont l'organisation est conçue. Dans le but de combler les lacunes signalées dans les définitions de l'organisation, Jacques Ferber a proposé un modèle organisationnel dans lequel l'organisation est partitionnée en groupes de rôles et les relations entre les groupes ont été bien exhibées. Ainsi, le modèle Agent-Group-Rôle (AGR) a été proposé pour la première fois dans (Ferber & Gutknecht, 1998), et puis il a été amplement exploité dans la littérature spécialisée comme : (Ferber, Michel, & Baez-Barranco , 2004), (Ferber, Gutknecht, & Michel, 2004), (Hoogendoorn & Treur, 2009), (Chebout, Mokhati, Badri, & Babahenini, 2016) et (Laouadi, Mokhati, & Seridi-Bouchelaghem, 2017).

Dans le contexte de notre projet de thèse, nous optons pour le modèle AGR pour l'implémentation des SMA ouverts vu les spécificités qu'il procure et également sa

capacité de modéliser les caractéristiques inhérentes aux SMA ouverts telles que l'ouverture et l'hétérogénéité. En fait, l'architecture interne (hétérogénéité au niveau agent) des agents dans le modèle AGR n'est pas spécifiée et est laissée à la charge des développeurs. Cela signifie que la formulation d'un SMA ouvert basé-AGR est complètement hétérogène. De plus, la constitution des groupes d'agents n'est pas transparente. En d'autres termes, les groupes dans le modèle AGR sont considérés comme des boites noires, où ce qui se passe à l'intérieur d'un groupe donné n'est plus accessible que par les agents qui composent ce groupe (hétérogénéité au niveau groupe). Dans le modèle AGR, un agent peut jouer, également, un ou plusieurs rôles et réciproquement un rôle peut être joué par un ou plusieurs agents sachant que les agents ne peuvent communiquer que s'ils appartiennent au même groupe.

Les SMA ouverts basés sur des modèles organisationnels ont été appliqués dans de nombreux autres domaines tels que les institutions électroniques (IE) (Esteva, Rodríguez-Aguilar, Sierra, Garcia, & Arcos, 2001)(García-Camino, Noriega, & Rodríguez-Aguilar, 2005)(Arcos, Esteva, Noriega, Rodríguez-Aguilar, & Sierra, 2005), le e-commerce (Ferber, Gutknecht, & Michel, 2004), etc. C'est pourquoi l'étude des SMA ouverts a pris beaucoup d'envergure dans la recherche vu la croissance remarquable des systèmes informatiques (ou encore l'informatique ubiquitaire) d'un côté, et la migration des anciens systèmes informatiques (legacy systems *en anglais*) vers des solutions ouvertes et extensibles où les besoins fonctionnels sont exprimés par des composants logiciels additifs d'un autre côté.

Par ailleurs, l'activité de contrôle, dans l'ingénierie des SMA, représente une tâche primordiale pour un système ouvert. De même, contrôler un système consiste à assurer, par le biais d'un processus de contrôle, l'achèvement de son but. On peut décider, dans ce cas que le système en question a assuré sa contrôlabilité. Ainsi, un SMA ouvert arrivera à achever le but pour lequel il est conçu si est seulement si sa contrôlabilité est vérifiée. Ceci mettre en exergue une question importante : « *comment guider le comportement d'un SMA ouvert vers un état cible à partir de n'importe quel état initial donné ?* » ou encore « *comment assurer la contrôlabilité pour un SMA ouvert ?* ».

Les SMA ouverts doivent être augmentés avec des normes avec lesquelles le système est soumis à des contraintes comportementales. Les normes informatiques sont, en fait, inspirées des sciences sociales où les normes sont habituellement rencontrées dans la vie quotidienne. Ainsi, les normes ont été introduites initialement dans les SMA ouverts comme une nécessité pour guider le comportement des agents. Pendant les deux dernières décennies, les normes sont amplement utilisées dans la littérature dans le but majeur de contrôler particulièrement le comportement des SMA ouverts en dépit des autres types de SMA (réactifs, holoniques, … etc.). Par définition, la notion de norme signifie le moyen avec lequel on peut contrôler le comportement des agents au sein d'un SMA (Rotolo & Van der Torre, 2011), (Criado, Argente, Noriega, &

Botti, 2013) et (da Cunha, Tassio, Marx, & Lucena, 2018). De plus, un system normatif est défini, d'après (Meyer, 1993), comme : « *systems in the behavior of which norms play a role and which need normative concepts in order to be described or specified* ». C'est ainsi que les systèmes normatifs sont liés intimement avec la logique déontique (Wright, 1951), (Meyer & Wieringa, 1994) et (Woleński, 2016) dans le sens où les normes sont exprimées en termes d'obligations, de permissions, d'interdictions et de recommandations (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014). Un autre détail ajouté par (Meyer, 1993) concernant les modalités déontiques precise : « *Deontic logic provides a means … by using special modal operators that indicate the status of behavior: that is whether it is legal (normative) or not* ». Autrement dit, les normes consistent pour un agent de savoir quelle est l'action permise (ou recommandée), obligée ou interdite à entreprendre. Par conséquent, les SMA normatifs représentent un exemple concret de l'utilisation de la théorie sociale dans les SMA, ou encore la relation entre la théorie de l'agent et les sciences sociales comme : la sociologie, la philosophie, l'économie et les sciences juridiques (Boella, Torre, & Verhagen, 2006).

Dans la théorie de contrôle (Kálmán, Falb, & Arbib, 1969)(Zabczyk, 1995) : un processus de contrôle est divisé essentiellement en deux parties: l'observabilité (ou le monitoring) et le contrôle proprement dit. Le monitoring représente la phase préliminaire qui précède tout processus de contrôle. Par définition, un moniteur est un outil logiciel qui s'exécute parallèlement avec un autre système dont l'objectif est de fournir des informations nécessaires sur le déroulement de l'exécution du système étudié (ISO/IEC/IEEE 24765, 2017). Dans le contexte des SMA, un moniteur consiste à récupérer des informations pertinentes à savoir des informations sur le cycle de vie d'un agent, la communication des agents, les rôles joués par un agent à un instant donné, etc. (Doan Van Bien, Lillis, & Collier, 2010a), (Chebout, Mokhati, & Badri, 2015). Par conséquent, Santag (Sontag, 2013) a défini le concept de la contrôlabilité comme étant la capacité de déplacer l'état d'un système donné avec toute sa configuration actuelle en utilisant uniquement quelques manipulations potentielles (les normes).

2. Contributions

Dans la littérature, il existe plusieurs travaux qui ont étudié le contrôle des SMA ouverts. Nous citons à titre d'exemple dans la gamme des travaux traitants le monitoring des SMA : (Doan Van Bien, Lillis, & Collier, 2010a), (Doan Van Bien, Lillis, & Collier, 2010b), (Esteva M. , Rosell , Rodriguez-Aguilar , & Arcos, 2004), (Mani, Garousi, & Far, 2008), (Nwana, Ndumu, Lee, & Collis, 1999) et (SnifferJade). D'autre part, nous citons dans les travaux qui concernent le contrôle : (García-Camino, Noriega, & Rodríguez-Aguilar, 2005), (Gonzalez-Palacios & Luck, 2006),(Hexmoor, Gunnu-Venkata, & Hayes, 2006), (Boella & Torre, 2008), (Hollander & Wu, 2011), (Criado, Argente, Noriega, & Botti, 2013) et (da Cunha, Tassio, Marx, & Lucena, 2018). Bien que ces approches aient apporté des réponses intéressantes aux différents

problèmes relatifs au contrôle des SMA ouverts, ils ne montrent pas clairement comment ces systèmes ont été implémentés. Par ailleurs, ils ne prennent plus en considération les spécificités des SMA ouverts basés sur des modèles organisationnels notamment l'hétérogénéité (au niveau agent et au niveau groupe), la spécification claire des mouvements des agents en termes d'entrée et de départ, le cycle de vie des agents et la manière dont les agents communiquent particulièrement dans ce type de systèmes. La plupart des travaux traitent les SMA ouverts comme étant des systèmes dont l'hétérogénéité, le non-déterminisme, l'émergence sont des propriétés inhérentes, mais aucune clarification n'a été offerte pour justifier une de ces propriétés. Cependant, après avoir examiné la littérature spécialisée, nous pouvons justifier que le modèle AGR concrétise plusieurs propriétés attachées à l'ouverture. Par ailleurs, l'hétérogénéité est clairement représentée dans le modèle AGR par le fait que la structure des agents et des groupes n'est plus spécifiée. De plus, la majorité des travaux sur le monitoring ne prennent pas en compte le facteur temps et la modélisation dynamique du comportement du système.

Une première synthèse de notre problématique ainsi que des perspectives ont été présentées dans la conférence internationale ICINCO en 2016 (Chebout, Mokhati, Badri, & Babahenini, 2016). Nous avons proposé, également, deux approches complémentaires pour le monitoring (Chebout, Mokhati, Badri , & Babahenini, 2019) et le contrôle[1] des SMA ouverts basés-AGR. Les contributions proposées dans le cadre de cette thèse sont organisées selon deux axes :

- **Le monitoring des SMA ouverts basés-AGR** (Chebout, Mokhati, Badri , & Babahenini, 2019): dans cette contribution, nous avons focalisé sur le monitoring des SMA ouverts basés-AGR dans le but d'observer, d'examiner et d'analyser dynamiquement le comportement d'un SMA ouvert pendant son exécution. L'approche proposée utilise les techniques de la Programmation Orientée Aspect (POA) pour le profilage et l'instrumentation. Par la suite, maintes informations pertinentes ont été récoltées à savoir : des informations sur le cycle de vie et les actions effectuées par les agents, la communication entre les agents, les groupes créés et détruits, les rôles accordés et rejetés, etc. Une modélisation dynamique des interactions effectuées en temps réel a été générée en conséquence. L'approche proposée est supportée par un outil logiciel et est validée par une étude de cas concrète.

- **Le contrôle des SMA ouverts basés-AGR :** dans l'objectif de profiter des résultats offerts par l'approche de monitoring précédemment discutée, une deuxième approche a été proposée dans le but d'influencer, à travers des moyens d'action, le comportement du SMA. Les moyens d'action utilisés dans

[1] *L'approche de contrôle proposée fait l'objet d'un article en état de soumission dans une revue internationale de renommée.*

l'approche de contrôle sont exprimés en termes de normes. À cet effet, nous avons contribué avec une nouvelle approche de contrôle basée sur les normes pour guider le comportement des SMA ouverts basés-AGR. Notre proposition est supportée par un outil logiciel et est validée, également, sur une étude de cas de notre implémentation.

3. Plan de la thèse

La présente thèse est organisée en cinq chapitres, les trois premiers chapitres sont consacrés à la présentation d'un état de l'art sur les travaux en liaison avec notre problématique. Ensuite, nos contributions seront présentées dans les deux derniers chapitres :

- **Systèmes multi-agents:** dans ce chapitre des concepts d'ordre général seront présentés autour de l'agent et des caractéristiques inhérentes aux SMA. Nous avons étudié les SMA ouverts d'un point de vue organisationnel en traitant les différents modèles organisationnels existants. Ensuite, nous mettons l'accent sur les données d'un problème de contrôle pour un SMA ouvert et finalement nous exposerons quelques plateformes agents basées sur des modèles organisationnels.

- **Normes et systèmes multi-agents normatifs :** dans le deuxième chapitre, nous allons présenter les différents concepts liés aux aspects normatifs dans les SMA ouverts à savoir : les architectures normatives dédiées aux SMA, la modélisation normative, les mécanismes de renforcement des normes et finalement la représentation et l'implémentation des normes. Un état de l'art sera élaboré et discuté à la fin de ce chapitre regroupant les travaux les plus pertinents.

- **Monitoring des systèmes multi-agents ouverts :** dans ce chapitre nous allons présenter des concepts clés autour du monitoring des SMA ouverts et les différents techniques utilisés comme le profilage et l'instrumentation. Ainsi, nous allons introduire le langage AspectJ avec ses constructions de base et les différentes techniques d'instrumentation et de profilage basées AspectJ. Finalement, nous allons projeter les techniques de monitoring sur les caractéristiques inhérentes aux SMA ouverts. Une étude comparative sera exposée et examinée à la fin de ce chapitre.

- **Une approche de monitoring des systèmes multi-agents ouverts :** dans ce chapitre une approche de monitoring des SMA ouverts basés-AGR sera présentée. L'approche proposée est supportée par un outil logiciel baptisé : RT-MTOMAS pour RealTime-Monitoring Tool for Open Multi-Agent Systems. RT-MTOMAS profite vivement des techniques de la POA pour l'instrumentation du code source. L'architecture de RT-MTOMAS sera

présentée en détail. Ainsi, l'outil RT-MTOMAS est validé avec une étude de cas nommé *travel-agency* sous la plateforme MaDKit. Finalement, une synthèse bibliographique a été mise en exergue pour discuter les points forts et les limites de notre proposition.

- **Une approche de contrôlabilité des systèmes multi-agents ouverts :** Le dernier chapitre est consacré pour élucider l'approche de contrôle proposée dans le cadre des travaux de cette thèse. Notre proposition est nommée NorCtrl4OMAS pour : Norms-based Control for Open Multi-Agent Systems. NorCtrl4OMAS utilise le langage JESS (JAVA Expert System Shell) pour l'implémentation des normes. Le choix d'utiliser JESS est justifié par la capacité de raisonnement sur les normes dont la norme la plus adéquate au contexte d'exécution sera instanciée. Le monitoring des normes, dans NorCtrl4OMAS, est effectué, également, en profitant des avantages de la POA. Notre proposition est supportée par un outil logiciel validé sur une étude de cas sous la plateforme MaDKit. Un bilan récapitulatif est dressé à la fin du chapitre pour bien montrer l'apport de NorCtrl4OMAS par rapport aux solutions existantes.

Chapitre 1

SYSTÈMES MULTI-AGENTS

Sommaire

1. Introduction

Les systèmes multi-agents (SMA) forment la branche de l'Intelligence Artificielle Distribuée (IAD) qui s'intéresse à la mise en interaction d'un ensemble d'agents dans le but de résoudre un problème donné. À cet effet, la résolution d'un problème quelconque nécessite, dans la majorité des cas, l'intervention des entités externes munies avec de nouvelles fonctionnalités requises ou de procéder à une modification des entités existantes (Ferber, 1995) (Vercouter, 2000). Ainsi, étendre ou mettre à jour les fonctionnalités d'un système nous conduit à identifier un nouveau concept inhérent aux SMA connu sous l'*ouverture.* Ainsi, ouvrir un SMA consiste à le rendre accessible par son environnement dans lequel des agents externes peuvent entrer et quitter le système à tout moment. L'ouverture par conséquent peut provoquer pour un SMA des perturbations, parfois inévitables, qui peuvent conduire à l'émergence de comportements non souhaitables.

Dans ce chapitre, nous commençons par introduire les SMA et présenter ses concepts de base, ensuite nous mettons l'accent sur le concept de l'organisation et comment peut-on implémenter un SMA ouvert en se basant sur un modèle organisationnel. Par ailleurs, nous étudions le sujet de l'affectation des rôles dans un SMA ouvert et les célèbres plateformes agents basées sur des modèles organisationnels. Afin de bien exposer notre problématique, nous présentons les données d'un problème de contrôle pour un SMA ouvert.

2. Concept d'agent

Malgré les efforts de standardisation des aspects opérationnels et techniques notamment de la FIPA (FIPA, 2000), la notion d'agent reste encore controversée concernant sa définition. À cet effet, plusieurs définitions ont été proposées au terme *agent* selon plusieurs points de vue (intelligence artificielle (IA), philosophie, robotique, etc.). En IA, on parle d'une entité intelligente qui présente un comportement intelligent (Wooldridge & Jennings, 1995). En philosophie, le terme *agent* est employé pour éluder l'utilisation du mot humain dans les différents modèles philosophiques exprimant une image abstraite du monde réel (Stratulat, 2002). En informatique, cependant, les caractéristiques qui font de l'agent un nouveau paradigme, dont une nouvelle manière de penser a été imposée, suscitent beaucoup de polémiques, notamment en raison d'une utilisation abusive. À titre d'exemple, dans le jargon des systèmes d'exploitation le terme *agent* peut être considéré comme un processus à synchroniser dans certains algorithmes qui utilisent le calcul distribué ou parallèle. Par ailleurs, la firme Microsoft (Microsoft, 2018) utilise la notion *d'agent* pour décrire un composant logiciel qui a pour vocation l'assistance technique des utilisateurs humains dans l'utilisation des plateformes logicielles. De même, *internet softbot* (software robot) est une solution de type IA

déployée sous forme d'un agent permettant la mise en interaction de plusieurs ressources sur internet à savoir : fichiers, e-mail, etc. (Etzioni & Weld, 1994).

En revanche, dans le domaine des SMA, l'une des définitions qui fait un consensus est donnée par Jacques Ferber (Ferber, 1995)(Ferber, 2006) :

> « *Un agent est défini comme une entité physique ou virtuelle, autonome, capable d'agir dans un environnement, de communiquer avec d'autres agents, de percevoir son environnement et de se reproduire (éventuellement)* »

Selon l'utilisation du terme agent, Michael Wooldridge fait la distinction entre deux notions, à savoir : la notion faible et la notion forte de l'agent (Wooldridge & Jennings, 1995) :

- *La notion faible de l'agent* : dénote l'agent comme étant une entité matérielle ou logicielle qui se caractérise par les propriétés suivantes :
  - *L'autonomie* : désigne la capacité de l'agent de se comporter individuellement sans intervention externe (humaine ou autre) et d'avoir un auto-contrôle sur ses actions et son état interne.
  - *La réactivité* : la capacité de l'agent de percevoir son environnement et répondre aux différents changements apparus dans le temps adéquat.
  - *La sociabilité* : la capacité d'interagir avec les autres agents par le biais d'un langage de communication.
  - *La pro-activité* : la capacité pour un agent de prendre l'initiative en montrant un comportement dirigé par ses objectifs.
- *La notion forte de l'agent* : en plus des propriétés précédentes, cette notion consiste à enrichir l'agent par des caractéristiques issues de l'être humain. Par exemple : l'utilisation des notions mentalistes spécifiées en termes de connaissances, buts, intentions et obligation (Shoham, 1993). De manière similaire, certains chercheurs sont allés plus loin et considèrent des agents *émotionnels* (Bates, 1994)(Bates, Loyall, & Reilly, 1994). Finalement, il est intéressant de penser à concevoir les agents en se basant sur des états mentaux inhérents à l'être humain (human-like mental states).

Wooldridge a ajouté, également, aux caractéristiques précédentes les propriétés suivantes :

- *la situation* (Wooldridge, 2002)(Wooldridge, 2009) : la situation d'un agent désigne l'environnement dans lequel un agent se situe et interagit. Ferber (Ferber, 1995) définit l'environnement par l'ensemble des objets passifs composant l'entourage de l'agent, à savoir : une base de données, une ressource matérielle, etc. Ainsi, l'environnement est considéré comme une entité active et dynamique (Guessoum & Mandiau, 2012).

- *La mobilité* : désigne la capacité de se déplacer d'une plateforme à une autre dans un réseau (White, 1994).
- *La véracité* : l'hypothèse qu'un agent ne communique pas des informations falsifiées (Galliers , 1988).
- *La bienveillance* : l'hypothèse qu'un agent ne possède pas des buts contradictoires et par conséquent chaque agent essaye d'accomplir la tâche qui lui ai affectée (Rosenschein & Genesereth, 1985).
- *La rationalité* : l'hypothèse qu'un agent se comporte dans le but de réaliser ses buts et également il ne se comporte plus de manière à interdire ses buts d'être achevés (Galliers , 1988).

La figure suivante (figure 1.1) résume les caractéristiques précédemment discutées :

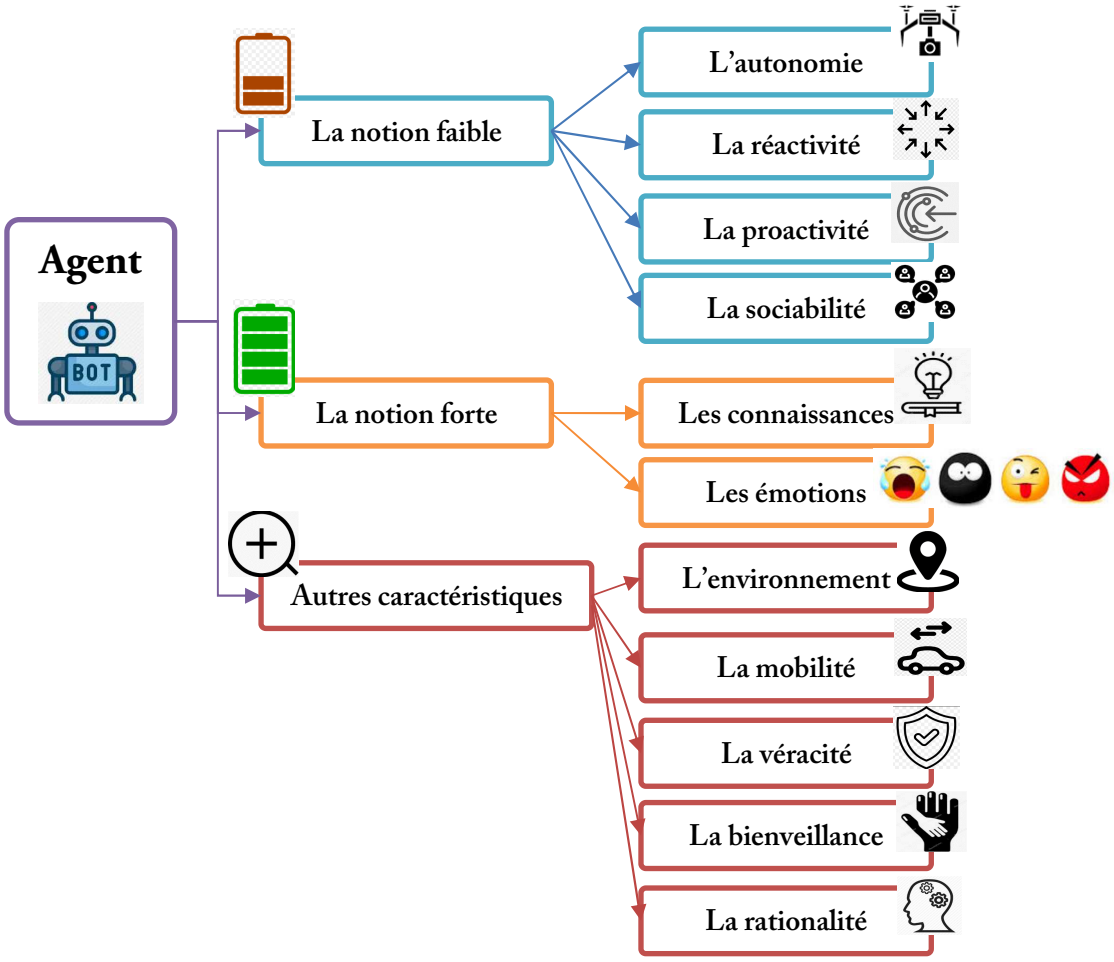


Figure 1.1 : Caractéristiques de l'agent selon Wooldridge.

3. Systèmes multi-agents

Un système multi-agent (SMA) est composé par un ensemble d'agents. Les agents au sein d'un SMA interagissent dans le but de réaliser un but commun ou pour atteindre des buts différents. Ainsi, l'achèvement d'un but central pour un SMA se réalise, certainement, par l'interaction et la collaboration des agents. Nous allons nous

focaliser pour la définition d'un SMA sur les définitions de Jacques Ferber et Yves Demazeau.

Selon Ferber (Ferber, 1995), un SMA est défini par :

- Un Environnement **E :** un espace disposant d'une métrique.
- Un ensemble d'objets **O** situés : pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.
- Un ensemble **A** d'agents qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.
- Un ensemble de relations **R** qui unissent des objets (et donc des agents) entre eux.
- Un ensemble d'opérations **Op** permettant aux agents A de percevoir, produire, consommer, transformer et manipuler des objets de O.
- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

En revanche, Demazeau (Demazeau, 1995) propose une décomposition du SMA, nommé l'approche voyelle, selon quatre dimensions qui correspondent aux voyelles : A, O, I et E pour : Agent, Organisation, Interaction et Environnement respectivement :

- *Agent* : représente le noyau de base qui compose un SMA.
- *Organisation* : structurer une société d'agents par des relations entre agents ou par la décomposition d'une tâche globale du système qu'elle assigne aux agents (Vercouter, 2000).
- *Interaction* : désigne la mise en communication des agents en se basant sur un langage de communication. Ainsi, l'interaction représente l'influence d'un agent sur le comportement des autres agents à l'aide d'une série d'actions (Ferber, 1995).
- *Environnement* : désigne tout ce qui entoure l'agent, à savoir : des objets physiques (i.e. environnement physique) ou d'autres agents (i.e. environnement social) (Ferber, 1995).

L'approche VOYELLE a été étendue en ajoutant la cinquième voyelle U pour décrire l'utilisateur. En fait, les agents au sein d'un SMA doivent prendre en considération les commandes émises par les utilisateurs humains et maintiennent, en conséquence, une représentation des utilisateurs regroupant des connaissances et croyances nécessaires à leur fonction. Cette représentation est connue souvent sous le nom *profil utilisateur* (Vercouter, 2000).

4. Organisation dans les SMA

Depuis leur apparition dans les années 80, les SMA ont été considérés comme des sociétés d'agent. Ainsi, les agents dans une société interagissent ensemble et coordonnent leurs comportements dans le but de réaliser un but global. À cet effet, un SMA peut être étudié d'un point de vue individuel (niveau micro) en mettant l'accent sur l'agent. On parle dans ce cas des SMA Centrés-Agent (SMACA). Il peut également être étudié d'un point de vue social (niveau macro), on parle dans ce cas des SMA Centrés-Organisation (SMACO) (Mansour, 2007)(Mansour & Ferber, 2007).

Les SMACA désignent les SMA classiques dont les recherches se focalisent principalement sur l'agent. Par conséquent, les architectures proposées autour les SMACA mettent l'agent au centre de leurs préoccupations. Ainsi, les approches centrées agents étudient l'agent selon deux angles distincts : le fonctionnement interne et le comportement externe (Yoo & Briot, 1999). Le fonctionnement interne désigne les caractéristiques de base de l'agent telles que : l'autonomie, la réactivité, la pro-activité, la planification, etc. Cependant, le comportement externe étudie la relation de l'agent avec son environnement et avec les autres agents (accointance) (Mansour, 2007). Par ailleurs, les SMACA souffrent de quelques lacunes lors de l'ingénierie des systèmes à grande envergure. Quelques inconvénients ont été soulignés dans (Jennings & Wooldridge, 2000):

> *"Another common misconception is that agent-based systems require no real structure. While this may be true in certain cases, most agent systems require considerably more system-level engineering than this. Some way of structuring the society is typically needed to reduce the system's complexity, to increase the system's efficiency, and to more accurately model the problem being tackled".*

D'après (Jennings, 2000), cela conduit à deux inconvénients majeurs :

- Les résultats des interactions sont par nature : imprévisibles.
- Il est extrêmement difficile (parfois impossible) de prédire le comportement du système en fonction de ses composants constitutifs, en raison de la forte probabilité de comportement émergent (et indésirable).

L'architecture diversifiée des agents pose un problème sérieux et est considérée comme l'une des raisons principales contribuant à l'émergence. En fait, les agents au sein d'un SMA sont conçus et structurés par différents concepteurs et interagissent ensemble pour des objectifs qui peuvent être communs ou contradictoires. Ainsi, le problème de sécurité dans les SMACA s'impose. Cela est justifié par le fait que les agents communiquent librement et aucun moyen de contrôle n'a été mis à disposition. À cet effet, Jacques Ferber montre à travers un exemple cette situation : la liberté de communication ouvre la porte à un type d'agents intrus qui se

comportent comme étant des pirates et affectent la progression du système d'une manière frauduleuse (Ferber, Gutknecht, & Michel, 2004). Ceci dit, les agents doivent sélectionner prudemment leurs interlocuteurs.

Les problèmes précédemment discutés ont motivé les chercheurs pour leur faire face et à penser à augmenter les SMACA par des mesures organisationnelles. Jennings et al. ont proposé la redéfinition des SMACA en ajoutant un niveau social (Jennings, 2000). De manière similaire, Jacques Ferber et al. ont introduit différents concepts inhérents à la dimension organisationnelle ajoutée à savoir : Agent, Groupe, Rôle, Service et Environnement. Ainsi, le modèle organisationnel AGR (Agent/Group/Rôle) a été introduit pour la première fois en 1998, puis il a été largement étudié par les chercheurs (table 1.1). Avant de plonger dans les définitions et clarifications autour le modèle AGR et ses extensions, nous allons initialement bien cerner le terme organisation dans le monde des SMACO.

Afin de définir l'organisation dans le contexte des SMA, Ferber (Ferber, 1995) à l'instar des autres chercheurs a adopté la définition de Edgar Morin (Morin, 1977) :

> *« L'organisation peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus. »*

De même, Wooldridge et al. (Wooldridge, Jennings, & Kinny, 2000) ont proposé une autre définition :

> *« We view an organisation as a collection of roles, that stand in certain relationships to one another, and that take part in systematic institutionalised patterns of interactions with other roles. »*

Partant de ces définitions qui omettent la manière avec laquelle l'organisation est conçue est partitionnée, Ferber et al. (Ferber, Gutknecht, & Michel, 2004) ont augmenté la définition de l'organisation par les caractéristiques suivantes :

- Une organisation est constituée par un ensemble d'agents (individus) qui manifestent un comportement donné.
- Une organisation offre un moyen pour partitionner le système en groupes. Chaque groupe constitue un contexte d'interaction pour les agents. Les agents du même groupe peuvent communiquer librement à l'intérieur du même groupe. Ainsi, ce qui se passe à l'intérieur d'un groupe ne peut être accessible que par les agents qui appartiennent à ce groupe.
- Le niveau organisationnel décrit pour un SMACO le « Quoi ? » et non le « comment ? ». Ainsi, des schémas sur les activités des agents ont été imposés, mais ils ne spécifient pas la manière avec laquelle les agents se comportent. Autrement dit : Qui fait *quoi*, mais pas *comment* ?

- Concevoir un système avec la prise en compte du niveau organisationnel consiste à négliger les détails d'implémentation. Le choix d'implémenter un agent particulier pour jouer un rôle spécifique est laissé ouvert.

Ainsi, Ferber a fait la distinction entre deux niveaux de l'organisation à savoir : le niveau structurel et le niveau concret (Ferber, 1995)(Ferber, Gutknecht, & Michel, 2004). L'organisation structurelle désigne les caractéristiques abstraites d'une organisation, l'organisation nécessite donc un minimum de structuration afin que l'ensemble, formé par des membres à priori hétérogènes, soit cohérent (Ferber & Gutknecht, 1998). Par contre, une organisation concrète représente une instance de l'organisation structurelle. Par conséquent, l'étude d'une organisation doit se faire selon trois axes (Ferber, 1995) :

- *Un axe structurel* : conception d'une ossature organisationnelle au travers des groupes, rôles, interactions.
- *Un axe fonctionnel* : identification des fonctions (objectifs) des membres de l'organisation et les moyens pour les atteindre.
- *Un axe normatif* : déterminer les règles et les normes qui régiront l'activité et les interactions des agents au sein de l'organisation.

5. Modèle AGR et ses extensions

Le modèle AGR est basé sur trois concepts de base : Agent, Group et Rôle respectant un ensemble d'axiomes. D'après Olivier Gutknecht (Gutknecht, 2001), les concepts de base du modèle AGR sont définis comme suit :

- *Agent* : Une entité agissante, communicante, autonome, et dont l'action s'exerce dans un contexte social. Ferber et al. (Ferber, Gutknecht, & Michel, 2004) rajoutent d'autres contraintes : un agent joue un rôle dans un groupe et peut être membre d'un ou de plusieurs groupes.
- *Groupe* : un terme générique pour qualifier une communauté d'agents en relation (par interaction, par partage d'un environnement, par un but ou une ontologie commune, ...). Deux agents peuvent communiquer si et seulement s'ils sont membres du même groupe.
- *Rôle :* est une représentation abstraite d'une fonction du groupe pouvant contraindre le comportement de l'agent, et incarnée dans un ou des comportements spécifiques de l'entité. Un rôle peut être joué par un ou plusieurs agents. Un rôle, également, doit être demandé explicitement par l'agent.

Dans cette section, nous proposons un aperçu des différentes approches étendant le modèle AGR. Par ailleurs, il est intéressant de noter que plusieurs approches traitant le rôle dans les organisations des agents ont été proposées dans la littérature. Chacune de ces approches aborde le concept de rôle différemment. Par exemple, la

méthodologie GAIA (Wooldridge, Jennings, & Kinny, 2000) similairement au modèle AGR focalise sur l'aspect conceptuel du rôle. Tandis que, la méthodologie MOISE/MOISE+ (Hannoun, Sichman, & Sayettat, 1999) essaye de résoudre le problème de faisabilité lié à l'utilisation des rôles tel que la relation agent/rôle. La raison avec laquelle le rôle est considéré, dans la méthodologie MOISE/MOISE+, comme un ensemble de missions. Nous allons dans ce qui suit mettre l'accent, particulièrement, sur les modèles AGRS (AGR + Service) et AGRE (AGR + Environnement) :

5.1. AGRS (Mansour & Ferber, 2007)

Dans le modèle AGRS (figure 1.2) la distinction a été faite sur la manière d'exploiter le rôle à savoir : utiliser ou jouer un rôle. À cet effet, le rôle est défini comme étant un ensemble de services que l'agent doit fournir ou requérir. Par définition, jouer un rôle consiste pour un agent d'être inscrit comme joueur et fournisseur de services proposés par ce rôle. Tandis que, un utilisateur de rôle doit entrer en contact avec un agent joueur de rôle afin d'obtenir un ou plusieurs services.

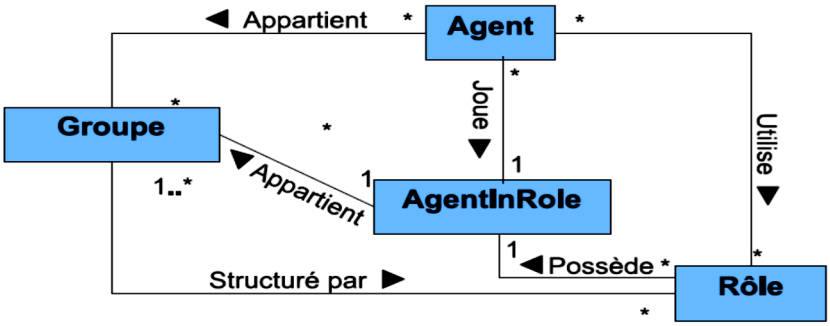


Figure 1.2 : Le modèle organisationnel AGRS (Mansour & Ferber, 2007).

Dans ce travail, les auteurs ont redéfini le concept de rôle (figure 1.3) qui est initialement laissé abstrait. Ainsi, un rôle est défini sous forme d'un sextuplé <Gr, Nm, Cd, Sv, At, Rl> où :

- Gr : identifiant du groupe,
- Nm : le nom du rôle.
  - Cd : l'ensemble de conditions du rôle à savoir :
  - CdJ : Les conditions que doit satisfaire un agent désirant jouer le rôle.
  - CdU : Les conditions que doit satisfaire un agent utilisateur du rôle.
- Sv : l'ensemble des services définis au niveau du rôle à savoir :
  - Les services offerts qu'utiliseront les agents utilisateurs du rôle.
  - Services fournis aux agents qui jouent le rôle. Tant que l'agent joue ce rôle, il peut utiliser ces services.
  - L'ensemble des services transférables que peuvent acquérir les agents joueurs et qui les garderont même quand ils lâchent le rôle.

- At : les attributs internes du rôle.
- Rl : l'ensemble des règles qui gèrent le rôle et sa dynamique. Elles utilisent les conditions et les attributs pour modifier l'état du rôle.

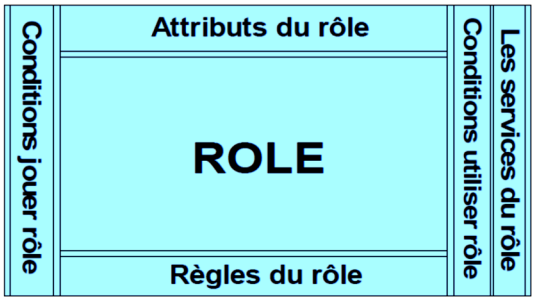


Figure 1.3 : La structure du rôle dans le modèle AGRS (Mansour & Ferber, 2007).

De même, la notion de groupe a été redéfinie de manière à tenir compte des groupes ouverts et dynamiques. Les groupes ouverts sont des groupes accueillant de nouveaux membres provenant des autres groupes ou de l'environnement. Tandis que les groupes dynamiques représentent les groupes dans lesquels la structure du groupe est modifiable dynamiquement. Par structure du groupe, on désigne les conditions d'entrée dans le groupe, l'ensemble des rôles joués et utilisés dans le groupe.

Ainsi, un groupe est défini par le tuplet <ID, Cds, Rol, Ser, Ats, Rls> où :

- ID : L'identifiant du groupe. Chaque groupe a un identifiant unique.
- Cds : Conditions d'appartenance au groupe.
- Rol : La liste des rôles que contient le groupe à un instant donné.
- Ser : L'ensemble de services offerts par les différents rôles du groupe.
- At : Les attributs internes du groupe.
- Rls : Les règles qui gèrent le groupe et sa dynamique. Elles utilisent les conditions et les attributs pour modifier l'état du groupe.

La structure du groupe dans le modèle AGRS est illustrée par la figure suivante (figure 1.4) :

Figure 1.4 : La structure du groupe dans le modèle AGRS (Mansour & Ferber, 2007).

5.2. AGRE (Ferber, Michel, & Baez-Barranco , 2004)

Dans le modèle AGRE, la notion de l'environnement a été ajoutée au modèle de base. L'environnement désigne à la fois les frontières physiques (ou géométriques) et sociales. Dans le contexte de AGRE, l'espace géométrique est nommé *zones* (areas) et l'environnement social est appelé *groupes* (au sens AGR). Les agents se situent dans des zones et effectuent des actions à travers des modes. Le mode décrit l'emplacement de l'agent et la façon dont il perçoit et agit dans un espace. Un mode dans une zone s'appelle un corps. Ainsi, un mode dans un groupe s'appelle un rôle. Un regroupement de zones s'appelle un monde. Le monde regroupe des zones de mêmes types. Dans le modèle AGRE, deux mondes ont été considérés (figure 1.5), *l'organisation* (l'environnement social) et le monde physique représentant l'environnement physique.

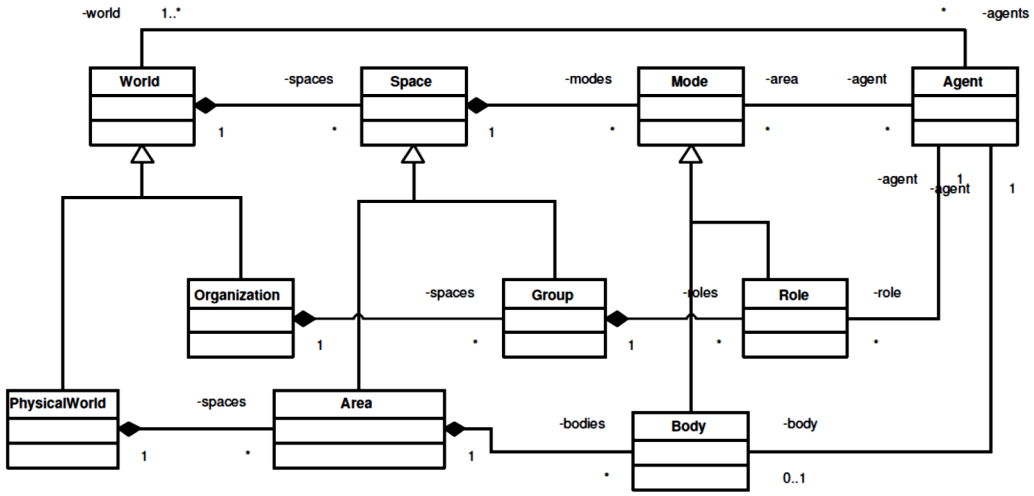


Figure 1.5 : Le méta-modèle du modèle AGRE (Ferber, Michel, & Baez-Barranco , 2004).

Il est intéressant de noter que les deux modèles AGRS et AGRE n'ont pas encore été concrétisés via des études de cas bien dédiées. Ainsi, la seule et unique plateforme agent qui incorpore le modèle AGR basique est MaDKit (Gutknecht & Ferber, 2000). Nous pensons que les recherches dans l'axe des modèles organisationnels utilisant AGR comme noyau sont encore vierges et nécessitent plus d'attention en termes d'implémentation. Les chercheurs peuvent s'inspirer des théories présentées dans les travaux précédents dans le but d'étendre la plateforme MaDKit pour qu'elle puisse supporter les modèles AGRS et AGRE. Finalement, la table suivante (table 1.1) résume les principales contributions autour du modèle AGR :

Travail	Contribution par rapport au modèle AGR
(Ferber & Gutknecht, 1998)	Le modèle AGR a été introduit pour la première fois.
(Ferber, Gutknecht, & Michel, 2004)	Un méta–modèle du modèle AGR a été proposé dont des concepts liés à l'organisation ont été ajoutés (i.e. MASCO)

(Ferber, Michel, & Baez-Barranco , 2004)	Le modèle AGR a été étendu avec le concept de l'environnement sous le nom : **AGRE** : AGR + Environnement (figure 1.5). Des concepts liés à l'environnement ont été introduits que ce soit l'environnement physique ou social.
(Mansour & Ferber, 2007)	Le modèle AGR a été étendu, également, par les concepts de service sous la dénomination **AGRS** : AGR + Service (figure 1.2). Dans ce travail le rôle a été exprimé en termes de services qu'un agent peut offrir ou requérir.
(Hoogendoorn & Treur, 2009)	Un modèle organisationnel adaptatif basé sur AGR a été proposé pour l'analyse et la conception des SMA basé organisation. Le modèle a été spécifié formellement et vérifié par le modèle checker.
(Laouadi, Mokhati, & Seridi-Bouchelaghem, 2017)	Une spécification formelle du modèle AGR de base a été proposée en se basant sur la logique de réécriture.

Table 1.1 : Recueil des travaux de recherches traitant le modèle AGR(Chebout, Mokhati, Badri , & Babahenini, 2019).

6. Systèmes Multi-Agents Ouverts

Pour qu'un SMA puisse être considéré comme ouvert, celui-ci doit respecter les propriétés d'un système ouvert (Vercouter, 2001). Un système multi-agent ouvert (SMA ouvert) se caractérise à la fois par son extensibilité (Sichman, 1995) et son évolutivité (Vercouter, 2001)(Vercouter, 2004). Ainsi, l'ouverture en termes d'extensibilité désigne la capacité d'ajouter et/ou de retirer dynamiquement dans le système des agents (Sichman, 1995), on parle dans ce cas d'un *SMA purement ouvert*. En revanche, l'ouverture en termes d'évolution permet d'enrichir des agents existants déjà dans le système par de nouveaux services, on parle dans ce cas sur un *SMA évolutif*.

L'extensibilité par conséquent change la composition du système en termes de nombre d'entités (agents) qui le constituent. L'ajout d'un nouvel agent permet d'augmenter le nombre d'agents dans le système (la taille du système) et réciproquement le retrait permet la décrémentation du nombre d'agents. Cependant, l'évolutivité permet non seulement de garder le même nombre d'agents, mais aussi de modifier le contenu d'un agent (Vercouter & Muller, 2010).

Ainsi, l'ouverture pour un SMA désigne, selon Boissier (Boissier, Gitton, & Glize, 2004), l'échange des informations avec l'extérieur. Les agents peuvent entrer et sortir du SMA ou encore être modifiés en cours d'évolution. Par ailleurs, Wooldridge définit un SMA ouvert d'un point de vue génie logiciel en mettant l'accent sur la composition du système pendant la phase de conception et d'implémentation. Pour Wooldridge, l'ouverture c'est l'impossibilité de savoir quels seront les composants d'un SMA ouvert ni comment ils vont interagir les uns et les autres (Wooldridge & Ciancarini, 2001) .

De plus, d'autres mesures ont été ajoutées à la notion des SMA ouverts, à savoir :

- L'hétérogénéité de ses composants (Criado, Argente, Noriega, & Botti, 2013),
- Le mouvement dynamique des agents (entrée et départ) pendant l'exécution du système (Felicíssimo, Lucena, Carvalho, & Paes, 2005),
- L'interaction des agents dans le but de réaliser chacun d'eux ses propres intérêts (Artikis, 2012)(Artikis, Sergot, Pitt, Busquets, & Riveret, 2016),
- Les interactions imprévues (Hewitt, 1991),
- L'absence d'une unité centrale (i.e. autorité) qui contrôle la globalité du système (Huynh, Jennings, & Shadbolt, 2006),
- L'aspect interne des agents n'est plus accessible, la seule et unique information sur les agents peut être captée à travers les messages échangés (Paes, et al., 2005),
- Les agents sont autonomes, hétérogènes et conçus indépendamment (par des tiers) et peuvent collaborer ensemble pour des buts similaires ou travailler séparément pour des fins contradictoires (Silva, Duran, Guedes, & Lucena, 2007),
- L'émergence du comportement global (Deguet, 2008),
- La multiplicité des centres de décisions (Vercouter, 2001),
- L'utilisateur joue un rôle primordial dans la perturbation du système (Boissier, Gitton, & Glize, 2004),
- Les agents jouant différents rôles dans un tel système augmentent l'opportunité pour des situations non souhaitables d'être présentes (Felicissimo, 2005),
- Le non-déterminisme qui provient du caractère ouvert du système où on ne peut plus prévoir l'état prochain du système à un instant donné, même pour un système clos et déterministe grâce à la complexité liée à l'émergence (Klein, 2009).

7. Affectation des rôles au sein des SMA ouverts

Partant de l'hypothèse que les SMA ouverts sont composés d'agents hétérogènes qui entrent et sortent dynamiquement à n'importe quel moment, le mouvement des agents au sein d'un SMA ouvert peut s'expliquer par le fait qu'un agent voulant accomplir une tâche donnée doit demander explicitement d'effectuer le rôle correspondant. En revanche, si l'agent vient d'atteindre son but il quittera la société en demandant, également, d'abandonner le rôle qui lui a été affecté.

Il est intéressant de bien clarifier le processus d'affectation du rôle donné à un agent demandeur. Autrement dit, quelles sont les conditions sous lesquelles l'agent peut jouer ou abandonner un rôle ou encore qu'elle est la signification pour un agent de jouer un rôle ? Le fait d'imposer des conditions pour un agent pour qu'il puisse jouer un rôle est motivé par sa nature hétérogène. Ainsi, les agents veulent atteindre leurs propres buts en contrepartie de l'objectif principal du système. Les buts individuels des agents peuvent être contradictoires et, par conséquent, la non-conformité de ses buts avec la spécification du SMA ouvert se produit (Artikis & Pitt, 2001).

À cet effet, Dastani et al. (Dastani, Dignum, & Dignum, 2003) ont imposé des critères judicieux pour la conception des SMA ouverts à savoir :

- Le besoin des cadres formels pour la spécification et la vérification de la structure des SMA ouverts et leurs buts d'une manière indépendante des agents participants,
- Le besoin des mécanismes avec lesquels les futurs agents participants peuvent évaluer les caractéristiques et les objectifs de la société,
- Munir les agents avec des outils pour leur permettre d'adapter leurs architectures et fonctionnalités avec les besoins fonctionnels des rôles qui leur sont affectés.

La discussion sur les rôles et le processus d'affectation des rôles au sein d'un SMA ouvert nous mène forcément à parler du modèle organisationnel choisi pour représenter la société d'agents. Dans (Dignum, Meyer, Weigand, & Dignum, 2002) trois modèles ont été introduits (figure 1.6), à savoir :

- Modèle Organisationnel (MO) : décrit le comportement et la structure globale de la société en termes de rôles, interactions et de normes sociales.
- Modèle Sociale (MS) : enrichi le modèle organisationnel par une population d'agents en leur assignant des rôles selon des critères définis dans des contrats sociaux. Dans ce modèle, les agents jouant effectivement des rôles sont appelés des acteurs.
- Modèle Interactionnel (MI) : spécifie les interactions entre les acteurs de la société.

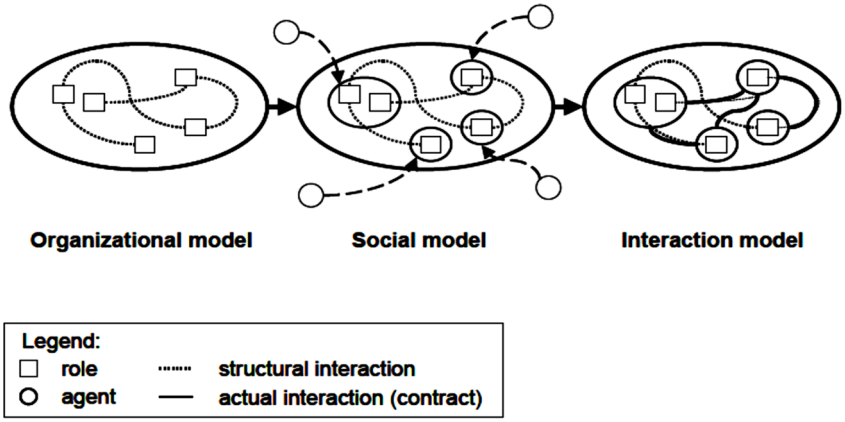


Figure 1.6 : Le modèle de la société d'agent (Dignum, Meyer, Weigand, & Dignum, 2002).

Selon (Dastani, Dignum, & Dignum, 2003), un rôle est défini comme une abstraction d'une fonction, service ou une politique. Ainsi, la description d'un rôle définit les activités et les services nécessaires pour atteindre le but de la société. De plus, une clarification des compétences nécessaires que doit avoir un agent pour accomplir un rôle donné doit avoir lieu. La spécification de rôle en terme de buts consiste à définir

les résultats que doit rechercher un agent jouant ce rôle, connu également comme les responsabilités du rôle tel qu'indiqué dans (Wooldridge, Jennings, & Kinny, 2000). Un rôle peut également être spécifié en termes de normes décrivant les obligations, les permissions et les interdictions imposées à l'agent jouant ce rôle. Finalement, la spécification des interactions entre les agents jouant les rôles décrit comment les agents doivent interagir les uns avec les autres.

En ce qui concerne la correspondance agent/rôle, Dastani et al. (Dastani, Dignum, & Dignum, 2003) définissent deux relations : *compatibilité* et *consistance* entre un agent et un rôle. Les relations proposées ont été basées sur la théorie des ensembles :

- Un agent est *compatible* avec un rôle si et seulement si : les buts (ou sous-buts) de l'agent représentent un sous-ensemble des buts (ou sous-buts) du rôle. Cela conduit naturellement à l'accomplissement des buts mentionnés dans le rôle. En revanche, un rôle est compatible avec un agent si et seulement si : les buts (ou sous-buts) du rôle représentent un sous-ensemble des buts (ou sous-buts) de l'agent.
  Exemple : dans un système d'évaluation des papiers dans une conférence ou une revue. Il y a d'une part, un agent A qui a le but de *lire un certain nombre de papiers* et il y a de l'autre part, le rôle d'un reviewer selon lequel un papier doit être examiné. L'agent A est *compatible* avec le rôle *reviewer* parce qu'un sous-but du rôle *reviewer* inclut la lecture du papier.
  La relation de compatibilité indique que l'agent est amplement souhaitable de jouer le rôle.
- La relation de consistance, cependant, indique que les buts de l'agent et du rôle ne doivent pas être en conflit.
  Exemple : un agent A avec un but de *soumettre un papier* dans une conférence est consistant avec le rôle *reviewer* parce que leurs buts ne s'interfèrent pas les uns avec les autres.

Pour que l'agent *A* joue le rôle *R*, il doit adopter les buts et les règles associés avec R en gardant ses propres buts (Dastani, Dignum, & Dignum, 2003). De plus, l'agent *A* doit inclure les obligations décrites dans le rôle dans sa base de connaissances. Pour atteindre son but, l'agent doit sélectionner un plan d'action qui correspond aux buts mentionnés dans le rôle qui lui ai affecté. Les buts sont ordonnés en priorités. Dans le cas ou *A* adopte les buts associés au rôle, il peut mettre à jour la priorité de ses propres buts en fonction des priorités des buts associés au rôle. Cela est faisable de plusieurs manières à savoir la nature de l'agent. S'il s'agit d'un agent égoïste (*selfish agent*), les buts de l'agent sont ordonnés en priorité. Sinon, les buts du rôle sont ordonnés en priorité dans le cas d'un agent social (*social agent*). Dans la majorité des cas, l'ordre des buts sera imposé par la société (Sichman & Conte, 1998) .

8. Données d'un problème de contrôle des SMA ouverts

Un SMA ouvert doit assurer, d'une part, certaines qualités dans le but d'atteindre un état cible souhaité, et d'exclure de l'autre part d'autres qualités. Ainsi, la fiabilité, la validation et la vérification doivent être respectées. Tandis que, l'hétérogénéité, le non-déterminisme, la dynamique, l'émergence et les changements sociaux provoquent des perturbations et par conséquent une instabilité qui l'amène à présenter un comportement non souhaité. Le contrôle, donc, a pour objectif de sortir de ce comportement pour revenir à la cible. Ainsi, l'ouverture sans contrôle peut conduire à des comportements chaotiques, comme mentionné dans (Esteva, Rosell, Rodriguez-Aguilar, & Arcos, 2004).

Dans la littérature, le problème de contrôle a été étudié largement par deux disciplines à savoir l'automatique et la théorie de contrôle (Klein, 2009). Le facteur commun entre ces deux disciplines est la manière de contrôler un système donné. Généralement, le contrôle se fait par imposition de contraintes sur le comportement du système dans l'objectif de le guider vers un état cible donné et éventuellement modélisé auparavant.

Dans ce contexte, Klein et al. (Klein, 2009) ont spécifié le problème de contrôle des SMA par trois démarches complémentaires à savoir, *la modélisation de la cible* ou de l'ensemble des cibles à atteindre par le système qui est considéré invariant dans le temps, la disposition des moyens d'observation dans le but d'avoir les connaissances nécessaires pour décider comment agir sur le système et également la disposition des moyens d'action pour influencer le comportement du système de manière à le diriger vers la cible. Pour le meilleur de nos connaissances, le moyen d'action le plus documenté et étudié dans la littérature est l'utilisation des normes (chapitre suivant). Un SMA normatif utilise conjointement les normes avec ses entités de base pour guider son comportement de manière appropriée.

9. Plateformes de développement des SMA ouverts

Une plateforme multi-agents consiste en une infrastructure logicielle comportant un ensemble d'outils, de bibliothèques, de services et de langages permettant la mise en œuvre des SMA. Ces outils peuvent également servir à la visualisation dynamique du comportement du système, l'analyse et le test des SMA ainsi créés.

Il existe un nombre important d'environnements de développement des applications orientées agent. Parmi les plateformes fournies comme logiciels libres on trouve : JADE (Bellifemine, Poggi, & Rimassa, 2001), ZEUS (Nwana, Ndumu, Lee, & Collis, 1999), SWARM (Minar, Burkhart, Langton, & Askenazi, 1996), MACE (Gasser, Braganza, & Herman, 1987), MaDKit (Gutknecht & Ferber, 2000) et Magentix2 (Alberola, Such, Botti, Espinosa, & Garcia-Fornes, 2013). Les deux dernières plateformes ont été connues par leurs capacités de développement des SMA qui sont

basés sur un modèle organisationnel, raison pour laquelle nous allons mettre une attention particulière sur ces deux plateformes. Ainsi, MaDKit et Magentix2 concrétisent maintes caractéristiques inhérentes aux SMA ouverts.

9.1. La plateforme Magentix2

La plateforme Magentix2 (Alberola, Such, Botti, Espinosa, & Garcia-Fornes, 2013)(Such, García-Fornes, Espinosa, & Bellver, 2013)(Magentix2) permet le développement des SMA ouverts dans lesquels des agents hétérogènes interagissent et s'organisent dans des organisations virtuelles (VOs : Virtual Organisations) (Foster, Kesselman, & Tuecke, 2001). Une organisation virtuelle représente un système ouvert formé par des groupes encapsulant des entités hétérogènes. Ces derniers collaborent dans le but de réaliser les objectifs de l'organisation. Magentix2 permet la gestion des SMA ouverts d'une manière sécurisée et optimale. Son objectif principal est d'apporter la technologie agent dans différents domaines à savoir : l'industrie, le e-commerce, la logistique, etc. Ainsi, Magentix2 est conçue dans l'objectif de développer des mécanismes robustes et efficaces pour le renforcement des normes afin de contrôler des applications complexes.

La plateforme Magentix2 a été conçue sur trois niveaux d'abstraction (figure 1.7) à savoir : le niveau organisationnel, le niveau interactionnel et le niveau agent.

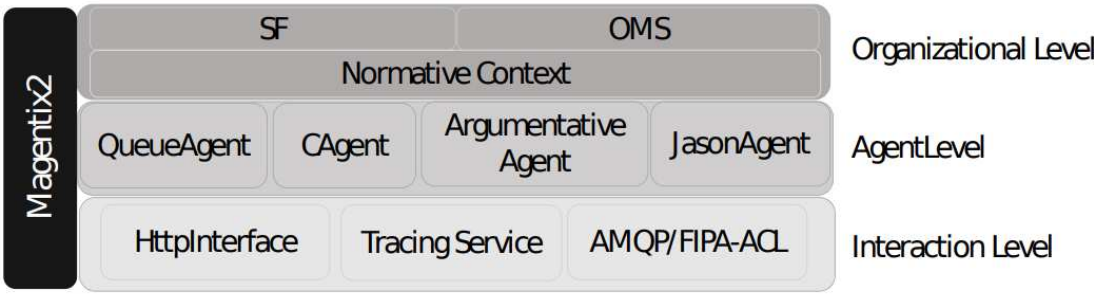


Figure 1.7: L'architecture de la plateforme Magentix2 (Valero, del Va, Alemany, & Botti, 2015)

- *Niveau organisationnel* : incorpore le cadre THOMAS (Argente, et al., 2011) pour la gestion des organisations virtuelles. THOMAS offre un ensemble de services modulaires dispatchés sur deux modules, SF (*Service Facilitator*) et OMS (*Organization Manager Service*). Le SF offre le service des pages jaunes/verts. Tandis que, le module OMS est responsable principalement de la gestion des organisations. Plusieurs types d'organisations existent. Une organisation peut être composée par d'autres organisations. De plus, il est possible de gérer les rôles pour chaque organisation. Un rôle est caractérisé par un ensemble d'attributs : sa position, son accessibilité et sa visibilité. Ces caractéristiques peuvent réduire l'accessibilité du rôle par l'OMS. Par ailleurs,

il est possible de définir des contextes normatifs pour restreindre l'accès aux services THOMAS.

- *Niveau interactionnel* : Magentix2 offre un protocole d'interaction flexible, des communications indirectes par le service de traçage et, également, des communications entre les organisations d'agents.
- *Niveau Agent* : dans ce niveau, plusieurs classes d'agents sont offertes pour les développeurs. Par exemple : CAgent (pour les conversations en parallèle) et JasonAgent (pour les agents BDI).

La plateforme Magentix2 apporte de plus la particularité de la gestion des aspects normatifs liés aux SMA ouverts par le biais de l'OMS. Cette dernière offre des services pour l'ajout et/ou la suppression des normes. Les services : *registerNorm* et *deregisterNorm* permettent aux agents de modifier les normes qui sont en vigueur dans le contexte de l'organisation. Ainsi, OMS permet la gestion des rôles et des groupes par le biais du service *informative service*. De plus, le service : *dynamic Service* permet aux agents de jouer et/ou abandonner des rôles à l'intérieur d'une organisation en utilisant les services : *acquireRole* et *leaveRole*. Cela montre, à l'instar de la plateforme MaDKit la gestion de l'ouverture exprimée en termes de mouvement des agents.

9.2. La plateforme MaDKit

MaDKit (Multi-agent Development Kit) est une plateforme de développement des SMA développée, initialement, par *Olivier Gutchneck* dans le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) dès 1998 (Ferber, 2009) (Gutknecht & Ferber, 2000). Ensuite *Fabien Michel* reprend le développement de la plateforme et y intègre, notamment, TurtleKit pour la simulation de type automate cellulaire.

La plateforme MaDKit est ancrée dans le modèle AGR de base (figure 1.2). La structure organisationnelle est donc implémentée au cœur de la plateforme MaDKit. Ainsi, MaDKit (figure 1.8) est basée sur trois principes (Gutknecht, 2001):

- Architecture à micro-noyau,
- Agentification systématique des services,
- Découplage fonctionnel entre noyau, agents et application d'accueil.

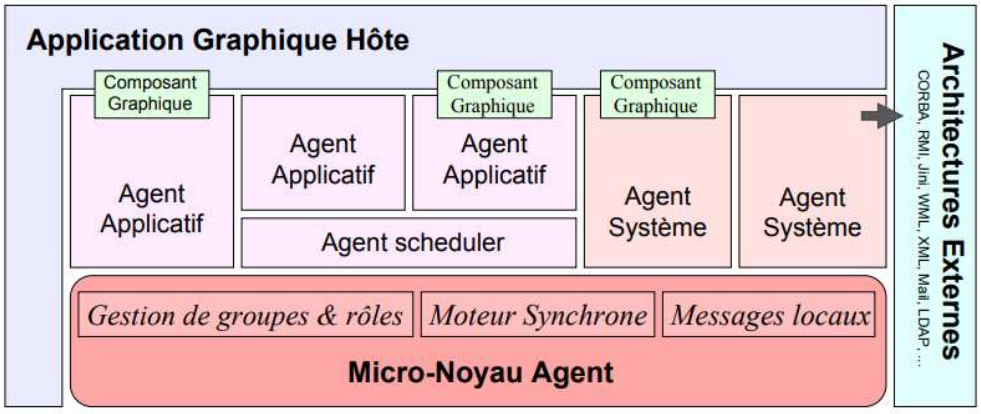


Figure 1.8 :L'architecture de la plateforme MaDKit (Gutknecht, 2001).

Concrètement, MaDKit est un ensemble de packages JAVA qui implémentent le noyau agent et diverses bibliothèques de base de messages, d'agents et de sondes.

MaDKit présente les qualités suivantes (Ferber, 2009):

- Simplicité de mise en œuvre et de prise en main ;
- Intégration très facile de la distribution au sein d'un réseau ;
- L'aspect pratique et l'utilisation efficace des concepts organisationnels pour créer différents types d'applications ;
- Hétérogénéité des applications et des types d'agents utilisables : on peut faire tourner sur MaDKit aussi bien des applications utilisant des agents réactifs simples de type fourmis que des applications disposant d'agents cognitifs sophistiqués.

En fait, la dernière qualité sur l'hétérogénéité des agents MaDKit montre clairement la valeur ajoutée par MaDKit par rapport aux plateformes de développement des SMA classiques où un modèle d'agent bien déterminé a été imposé. L'hétérogénéité au sens MaDKit désigne trois aspects fondamentaux, à savoir :

- *L'hétérogénéité des agents* : signifie la diversité des modèles et formalismes utilisés pour implémenter les agents.
- *L'hétérogénéité de communication* : les agents utilisent différents schémas d'interaction et de communication (communication par rôle, en mode broadcast, en mode point à point, en utilisant FIPA-ACL, etc.).
- *L'hétérogénéité applicative* : les SMA sont implémentés pour différents objectifs et dans plusieurs domaines.

Comme l'hétérogénéité est la propriété intrinsèque décrivant un SMA ouvert, nous apprécions positivement l'utilisation de MaDKit dans notre contexte de thèse en profitant des avantages et des diversités des techniques proposées. En plus de l'hétérogénéité, MaDKit offre un moyen pratique pour exprimer le mouvement des

agents en termes d'accès (ou de départ) au système. Ainsi, un agent qui veut jouer un rôle au sein de MaDKit doit passer explicitement une demande exprimée par la primitive *requestRole* et inversement, un agent désirant quitter le système doit exécuter la primitive *leaveRole*. Ceci dit, l'ouverture en terme de cardinalité (le nombre des entités formant le système) est assurée pleinement avec MaDKit. Les primitives *requestRole* et *leaveRole* à l'instar des autres primitives exprimant le mouvement des agents montrent clairement que la constitution du système change dynamiquement durant son exécution ce qui coïncide fortement avec les spécifiés des SMA ouverts. Il est intéressant de mentionner, également, que le terme ouverture signifie, pour un groupe d'agents ou une société, la permission de l'interaction avec l'extérieur (l'environnement) en recrutant de nouveaux agents occupant des rôles spécifiés dans le cadre des besoins fonctionnels nouvellement rencontrés.

Finalement, le choix de MaDKit est justifié, d'après ses créateurs, par le fait d'avoir une plateforme agent à la fois, générique, personnalisable, et évolutive d'une part et d'incorporer une couche logicielle extensible permettant la création des différents modèles d'agent et de services de base (Gutknecht & Ferber, 2000) d'autre part. Cependant, la version actuelle de MaDKit souffre de quelques lacunes concernant le monitoring des applications multi-agents. Ainsi, des informations spécifiques liées au contexte d'exécution ne peuvent pas être récoltées particulièrement des données en relation avec le modèle AGR et le cycle de vie des agents. Par ailleurs, des aspects relatifs à la normativité des SMA sont complètement omis dans MaDKit.

10. Conclusion

L'importance croissante du domaine des SMA comme étant un domaine de recherche fructueux vient du fait que ceux-ci offrent des solutions extensibles et évolutives à la fois aux problèmes fréquents. Étendre un SMA consiste à l'ouvrir à l'environnement dans lequel il se situe. L'ouverture pour un SMA désigne la capacité de changer la cardinalité du système par l'intégration de nouveaux membres occupant de nouvelles fonctionnalités. En revanche, la restructuration d'un SMA provoque des perturbations incontournables dans son comportement, ce qui génère des situations indésirables. À cet effet, il est intéressant de penser à doter un SMA ouvert par des mécanismes de contrôle afin qu'il puisse bien atteindre son objectif dans un délai bien précis.

Nous avons mis l'accent dans ce chapitre sur les concepts les plus pertinents à notre problématique à savoir les caractéristiques des SMA ouverts et les différents modèles organisationnels proposés pour implémenter un SMA ouvert. Ainsi, nous avons déterminé les données d'un problème de contrôle et les protocoles d'affectation des rôles au sein d'une société d'agents.

Dans le chapitre suivant, nous allons mettre au clair des concepts liés à la normativité des SMA en introduisant des concepts tels que : les normes avec leurs différentes

typologies et les SMA normatifs. Ainsi, un état de l'art sera élaboré à la fin du chapitre et couronné par une étude comparative des travaux traitant des normes dans les SMA.

Chapitre 2

NORMES ET SYSTÈMES MULTI-AGENTS NORMATIFS

Sommaire

1. Introduction

Dans ce chapitre, nous allons introduire des définitions liées au terme « norme » selon le point de vue de plusieurs disciplines traitant les normes. Les normes sont tellement étudiées dans la littérature de manière à restreindre et influencer le comportement des agents ou encore imposer des critères comportementaux dans le but d'atteindre un objectif bien déterminé. Ainsi, les travaux traitant le contrôle (ou la régularisation) des SMA par le moyen des normes sont proposés autour de plusieurs thématiques à savoir : *la création et la modification des normes*, *la représentation des normes*, *le cycle de vie de la norme*, *le renforcement des normes*, *l'émergence, l'internalisation et la transmission des normes et également le monitoring des normes*.

Nous débutons ce chapitre par des définitions autour du terme norme, ensuite nous abordons l'étude des systèmes multi-agents normatifs (SMA normatif), les types des normes et les moyens avec lesquels on peut modéliser une norme. Nous présentons en bref le langage JESS (JAVA Expert System Shell) comme étant le moyen le plus répandu dans la littérature pour représenter les normes, ensuite nous discutons le sujet de monitoring des normes et les mécanismes de renforcement des normes. Finalement, un état de l'art des travaux les plus pertinents sera élaboré à la fin de ce chapitre suivi d'un bilan récapitulatif montrant les lacunes des propositions discutées et les différentes directions à aborder.

2. Concepts et définitions

Le dictionnaire en ligne webster (merriam-webster, 2010) propose quatre définitions du terme *norme* à savoir :

1. Un standard autoritaire (provenant d'une source autoritaire).
2. Un principe pour une action juste contraignant les membres d'un groupe et servant à guider, contrôler ou règlementer un comportement correct et acceptable.
3. La moyenne, telle que :
    a. Un ensemble de standards dérivés généralement de la moyenne ou la médiane d'un grand groupe,
    b. Un motif ou un trait typique dans le comportement d'une société,
    c. Une pratique, procédure ou coutume répandue ou habituelle.
4. La norme géométrique, à savoir :
    a. La racine carrée de la somme des carrés des valeurs absolues des éléments d'une matrice ou un vecteur,
    b. La plus grande distance entre deux points successifs dans un nuage de points qui partitionne un intervalle donné en intervalles plus petits.

Dans l'encyclopédie britannique (Online Encyclopaedia Britannica , 2012), le terme norme est défini comme :

> *« a rule or standard of behaviour shared by members of a social group. Norms may be internalized; i.e., incorporated within the individual so that there is conformity without external rewards or punishments, or they may be enforced by positive or negative sanctions from without. Norms are more specific than values or ideals: honesty is a general value, but the rules defining what is honest behaviour in a particular situation are norms »*

Ainsi, dans la littérature, une myriade de définitions ont été proposées tel que mentionné dans (Hexmoor, Gunnu-Venkata, & Hayes, 2006): « *la norme a différentes définitions dans différents champs d'études comme les sciences sociales, la théorie des jeux, la psychologie et la théorie juridique* ». Nous citons à titre d'exemple :

- Une norme représente un comportement attendu par rapport à une situation spécifique (Ahmad, 2012).
- Le concept de norme est utilisé pour déterminer le comportement des agents au sein de la société et est accepté comme étant un moyen efficace pour normaliser leurs comportements (Alberti, Gomes, Gonçalves, Leite, & Slota, 2011).
- Les normes sont utilisées pour imposer des contraintes comportementales (Hollander & Wu, 2011).
- Les normes ont été promues en tant que mécanisme de coordination permettant de contrôler le comportement des agents (Criado, Argente, Noriega, & Botti, 2013).
- Être normatif représente le fait d'avoir un comportement conforme à la norme. En effet, il s'agit d'une caractérisation binaire, on l'est ou on ne l'est pas (Stratulat, 2002).
- Les normes permettent aux concepteurs de spécifier le comportement souhaité d'un système sociotechnique (Artikis, Sergot, Pitt, Busquets, & Riveret, 2016).
- La structure de la norme doit autoriser un certain degré de flexibilité permettant sa gestion en termes de : création, modification, suppression, activation et désactivation (Valero, del Va, Alemany, & Botti, 2015).
- Dans un système d'agents ouverts, la norme doit être communiquée aux agents nouvellement intégrés. Ainsi, le rôle de la norme est d'influencer le comportement d'un agent (Stratulat, 2002).
- Les normes sont introduites dans les SMA ouverts dans le but de contrôler l'autonomie des agents (Criado, 2013).

3. Système multi-agents normatifs

Bien que les SMA Normatifs aient vu le jour depuis deux décennies (Shoham & Tennenholtz, 1995)(Dignum, 1999), ils représentent une concrétisation de l'utilisation de la théorie sociale dans les SMA sachant que le terme norme a été extrait de la théorie sociale. Ainsi, un SMA Normatif, comme étant un axe de recherche, représente l'intersection des deux disciplines, les SMA et les systèmes

normatifs (Boella, Torre, & Verhagen, 2006). Le terme *normatif* désigne conforme ou basé sur la norme. Boella et al. (Boella & Torre, 2008) ont défini un SMA Normatif comme :

> *"A multi-agent system organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms, and to deliberate about norms and detect norm violation and fulfillment."*

Partant de cette définition, on constate que les normes représentent des mécanismes additifs apportés aux SMA classiques dans le but d'imposer un ordre organisationnel quelconque afin d'atteindre un objectif bien déterminé. À cet effet, les normes doivent être communiquées aux agents à titre informatif. Un SMA Normatif comporte toutes les entités logicielles et matérielles nécessaires pour la gestion des normes, à savoir, la transmission et le renforcement des normes. De même Meyer et al. (Meyer, 1993) ajoutent à la définition précédente ce qui suit :

> *" Systems in the behaviour of which norms play a role and which need normative concepts in order to be described or specified."*

Cela montre l'importance des normes comme étant un moyen pour spécifier et décrire le comportement d'un SMA. D'autre part, les agents dans un SMA Normatif interagissent au sein d'un environnement normatif de différentes manières. D'un point de vue agent, les agents peuvent créer, modifier, renforcer des normes en utilisant les rôles spécifiques définis dans un SMA Normatif comme les rôles de législateur, policiers, etc. Un législateur peut créer, activer, instancier, modifier ou désactiver une norme. Tandis que l'agent policier s'occupe du renforcement des normes en termes de sanctions ou récompenses. De manière similaire, un système normatif utilise des acteurs spécifiques (agents législateur et policier) pour maintenir l'équilibre dans un SMA. Par équilibre on désigne l'assurance d'un comportement idéal conduisant vers l'état cible souhaité. À cet effet (Boella, Torre, & Verhagen, 2006) distingue deux niveaux d'équilibre. Premièrement, les normes sont utilisées pour maintenir l'ordre social dans un SMA Normatif. Deuxièmement, les systèmes normatifs disposent des moyens pour se mettre à jour et s'adapter à l'évolution des changements dans l'environnement.

Un autre point intéressant à soulever concernant l'originalité de la norme, ou encore, la création de la norme ?. En fait, la norme ne doit pas nécessairement être promulguée par un législateur, mais plutôt elle peut être émergée spontanément ou être négociée par un groupe d'agents (Boella, Torre, & Verhagen, 2006). Ensuite, une autre question problématique devrait être posée sur la manière de renforcer la norme. Cela conduit effectivement à proposer des mécanismes de renforcement et de monitoring des normes. Nous allons mettre une attention particulière sur les aspects de monitoring et de renforcement des normes dans les sections suivantes vu la panoplie des recherches traitant ces cas.

Dans le but de projeter les aspects précédemment discutés sur les SMA ouverts, une autre perspective s'impose en conséquence de la nature hétérogène de ce type de systèmes. La cardinalité variée des SMA ouverts nécessite la proposition de nouveaux mécanismes de transmissions des normes aux agents nouvellement intégrés dans la société. Ainsi, dans un SMA ouvert les normes ne sont pas communiquées explicitement aux agents visiteurs (Savarimuthu, Purvis, Purvis, & Cranefield, 2009). Autrement dit, les aspects normatifs ne sont plus clairs pour les agents. Ces derniers, doivent être obligés de chercher la normativité liée à un comportement donné en utilisant des mécanismes de détection. Cependant, ce n'est plus le cas pour les agents autonomes où les aspects sociaux n'existent plus (ou partiellement). Ce type d'agents se comportent sans aucune intervention directe provenant de l'extérieur et disposent d'un moyen d'auto-contrôle sur les actions à entreprendre et leurs états internes. Dans ce contexte, plusieurs approches ont été proposées (voir section n° 9) dans le but d'inciter les agents à suivre la norme ou encore chercher à influencer les agents en les persuadant par le comportement normatif à travers des mécanismes de sanctions et de récompenses.

4. Typologies des normes

D'après (Coleman, 1998), les normes sont catégorisées en deux grandes familles : les normes conventionnelles et les normes essentielles. Les normes conventionnelles ou les conventions représentent les normes auxquelles on obéit non pas pour répondre à une attente sociale, mais plutôt pour des raisons morales ou rationnelles (Stratulat, 2002). Ainsi (Young, 1993), a défini les normes conventionnelles comme : "*un type de comportement habituel, attendu et auto-contraignant. Tout le monde se conforme et tout le monde s'attend à ce que les autres se conforment avec …*". On peut faire allusion aux coutumes et rites engendrés par la société. Cependant, les normes conventionnelles souffrent, parfois, d'un manque de pouvoir qui leur permet d'être renforcées par des obligations et des sanctions.

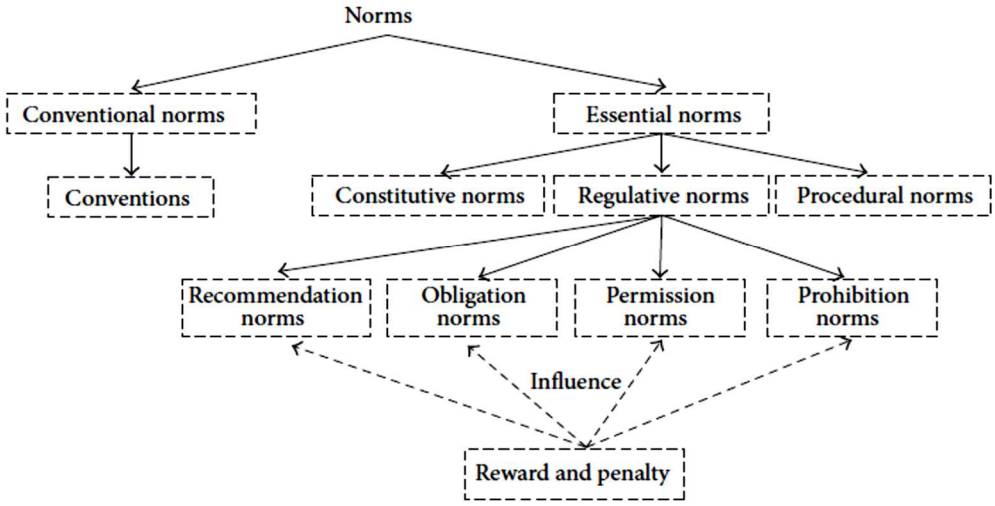


Figure 2.1 : La typologie des normes (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014).

En revanche, les normes essentielles ont pour objectif la résolution des problèmes provenant des conflits entre des intérêts individuels ou collectifs (Villatoro, Sen, & Sabater-Mir, 2010). À titre d'exemple : "la norme *ne pas polluer l'environnement urbain*" est essentielle, car elle oblige les individus de ne pas jeter leurs déchets à la place, mais plutôt de les transporter vers les endroits appropriés (Piskorski & Gorbatai, 2017). Par conséquent, tout le monde profite d'un tel comportement. Ainsi, les normes essentielles sont souvent associées à des sanctions.

La littérature des SMA normatifs suggère trois types de normes essentielles (figure 2.1) : les normes régulatrices, les normes constitutives et les normes procédurales. Les normes régulatrices désignent le comportement idéal que l'agent doit suivre par le biais des obligations, interdictions et permissions (Caire, 2007). Une norme régulatrice doit prescrire la sanction ou la récompense que doit subir un agent violant la norme (figure 2.1). Un autre type de norme régulatrice a été introduit pour la première fois dans (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014) sous le nom : recommandation qui désigne le fait de récompenser un agent effectuant une action permise. Les normes constitutives consistent à faire attacher les aspects non normatifs (aspects bruts) à ceux normatifs (Knobbout, 2016). Ainsi, les normes constitutionnelles sont utilisées principalement pour ajouter, supprimer ou mettre à jour des lois ou d'autres normes institutionnelles (Boella & Torre, 2004). Les normes institutionnelles, à la base, cherchent pour une institution (gouvernement ou association) à modifier le comportement de leurs individus. À cet effet, des mécanismes constitutifs ont été utilisés pour édicter les normes constitutives à savoir : le pouvoir social ou le vote. Le mécanisme de pouvoir social est réalisé par un agent disposant d'une autorité. Tandis que le mécanisme de vote est réalisé selon un protocole spécifique (Marir, Silem, Mokhati, Gherbi, & Bali, 2019). Boella et al. (Boella & Torre, 2006) ont différencié les normes régulatrices de celles constitutives à travers la notion de (*counts-as*) : si la norme régulatrice stipule que : *les véhicules sont interdits dans le parc*, la norme constitutive dit que : *les vélos sont également comptés comme des véhicules dans le parc*. Finalement, les normes procédurales sont des normes instrumentales adressées aux agents jouant des rôles dans les systèmes normatifs dans le but d'inciter ces agents à reconnaitre les violations ou à appliquer des sanctions. Ainsi, les normes procédurales ont pour objectif d'atteindre l'objectif de la société spécifié par les normes régulatrices et constitutives (merriam-webster, 2010). Par exemple : les normes procédurales expliquent comment un procès devrait être mené et quels sont les devoirs, les droits et les pouvoirs des juges, des avocats et des accusés (Boella & Torre, 2008).

5. Représentation des normes

Dans les systèmes normatifs, les normes utilisées doivent être représentées d'une manière qui leur permet d'être manipulées et traitées par les agents (Hollander & Wu, 2011). Selon (Savarimuthu , 2011), les chercheurs ont représenté les normes par

des structures de données implicites ou explicites. Ainsi, (Hollander & Wu, 2011) et (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014) ont référencé les quatre schémas de représentation les plus répandus à savoir : la logique déontique, les systèmes à base de règles, la théorie des jeux et les chaines binaires.

Nous allons, dans ce qui suit, expliquer chaque type de représentation des normes. Ainsi, le choix qui va mener à bien notre problématique de contrôle sera justifié à la fin de cette section.

5.1. La logique déontique

Historiquement, la logique déontique connue aujourd'hui apparait pour la première fois dans les travaux de George von Wright dans (Wright, 1951). Les travaux des logiciens dans le domaine de la logique déontique qui succèdent sont plus ou moins marqués par les idées de Wright présentées dans cet article. La logique déontique est développée de la logique classique qui représente une version étendue de la logique classique traitant les modalités de « nécessité » et « possibilité » (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014). De plus, la logique déontique représente une application de la logique à l'éthique et au domaine légal (Peterson, 2011). Ainsi, la logique déontique standard (SDL pour Standard Deontic Logic) comporte une unique modalité O qui dénote l'obligation. En fait, la logique déontique modale est construite en étendant la logique proportionnelle avec deux modalités abrégées : P et I qui désignent respectivement : la permission et l'interdiction.

C'est ainsi que l'opérateur O sera toujours considéré en tant que primitif, à l'aide duquel les opérateurs I et P sont définis :

- L'opérateur I peut être défini à partir de l'opérateur O comme :

$$I\alpha =_{def} O\neg\alpha$$

  Ainsi : *Interdit = Non permis = Obligatoire de ne pas.*

- L'opérateur P peut être défini à partir de l'opérateur O comme :
  - $P\alpha =_{def} \neg O \neg\alpha$

  Ainsi : *Permis = Non interdit = Non obligatoire de ne pas.*

Les prépositions Oα, Iα et Pα se lisent respectivement : α est *obligatoire* (ou il est obligatoire que α), α est *interdit* et α est *permis*.

Le système de logique modale contient tous les axiomes et toutes les règles d'inférence de la logique propositionnelle (Stratulat, 2002) à savoir :

- Le Modus Ponens MP: $\dfrac{\alpha \Rightarrow \beta \qquad \alpha}{\beta}$
- La nécessité NR: $\dfrac{\alpha}{O_\alpha}$

- L'axiome K : $$O(\alpha \Rightarrow \beta) \Rightarrow (O\alpha \Rightarrow O\beta)$$

Parmi les autres tautologies les plus utilisées, nous trouvons :

$$O\alpha \Rightarrow P\alpha$$

Autrement dit, l'obligation doit impliquer la permission correspondante.

Par ailleurs, un autre concept a été introduit dans (Stratulat, 2002) sous le nom : propriété déontique. « Une propriété indique si un agent est obligé, autorisé ou n'a pas le droit d'exécuter une action ou de réaliser l'apparition d'un état α». Ainsi, pour représenter les trois opérateurs déontiques O, I et P, on utilise les prédicats : O(Agent, α), I(Agent, α) et P(Agent, α).

Une norme peut avoir la forme suivante :

$$O^N(Agent, \alpha, \mathrm{i}) \leftarrow conditions$$

Avec :

- N : le représentant de l'autorité,
- α : une action,
- i : intervalle de temps,
- *conditions* : la condition sur le contexte d'application.

5.2. Les systèmes à base de règles

Les systèmes à base de règles sont composés par un ensemble des faits, de règles et un moteur d'inférence associé. Les faits représentent des portions de connaissance. L'ensemble des faits forme une base de connaissance. Les règles sont structurées sous la forme (Si P alors C) dont P et C sont appelés respectivement prémisse et conclusion. En logique de propositions ou de premier ordre, les règles sont décrites via des clauses contenant des propositions ou des prédicats (Hales, 2002)(Savarimuthu , 2011).

Trois types de raisonnement possibles dans un système à base de règles, à savoir : le chainage avant (raisonnement guidé par les données), le chainage arrière (raisonnement guidé par les buts) et le chainage mixte : utilise une combinaison des deux méthodes précédente.

Pour faire allusion à la normativité des SMA, le raisonnement dans les systèmes à base de règles permet de représenter les normes implicitement dans le processus de délibération des agents. De plus, une norme est implémentée comme étant une règle (García-Camino, Noriega, & Rodríguez-Aguilar, 2005).

Parmi les systèmes à base de règles les plus répandus dans la littérature, on trouve JESS (JAVA Expert System Shell) un langage de script proposé par (Sandia National

Laboratories, 2013) permettant l'écriture des systèmes experts. JESS offre un support pour doter des applications JAVA ordinaires par des moyens de raisonnement (i.e. incorporer le raisonnement dans des applications JAVA). Cela montre une très forte flexibilité en terme de manipulation du code JAVA à l'intérieur du code JESS et inversement. Nous allons élaborer dans la prochaine section un résumé sur les principaux constructeurs du langage JESS en mettant l'accent sur la représentation des normes particulièrement dédiée au modèle AGR.

5.3. Les chaines binaires

Les chaines binaires (*binary string*) sont des successions de zéros et de uns. Le caractère zéro représente l'absence de la norme et le caractère un représente l'occurrence de la norme. Ainsi, les chaines binaires sont utilisées amplement dans la recherche dans une population d'agents pour vérifier la transmission et l'émergence des normes (Caldas & Coelho, 1999)(Galán & Izquierdo, 2005). (Voir section n° 9, travail de Mahmoud et al.).

5.4. La théorie des jeux

La théorie des jeux est un domaine des mathématiques qui permet de comprendre formellement des situations dans lesquelles les joueurs (les preneurs de décision) interagissent. Un jeu est alors défini comme un univers dans lequel chaque preneur de décision possède un ensemble d'actions possibles déterminé par les règles du jeu (Bourlès & Henriet, 2017). D'après Laurent Vercouter (Vercouter, 2000), la théorie des jeux est par excellence un domaine compétitif où chaque participant agit selon son propre intérêt pouvant être incompatible avec les intérêts locaux de ses partenaires. Dans les SMA basés sur la théorie des jeux, les agents ont un ensemble d'actions parmi lesquelles ils doivent en choisir une à effectuer. Chaque agent est capable de faire un choix simple qui produit un gain correspondant (Hollander & Wu, 2011). À chaque tour de jeu, les agents tentent de maximiser leurs gains en choisissant une action en fonction de ce qu'ils anticipent que leurs adversaires choisissent. Les normes sont représentées par les stratégies qu'un agent utilise pour prendre des décisions (Mukherjee, Sen, & Airiau, 2007) (Savarimuthu, Purvis, Purvis, & Cranefield, 2009). Une norme émerge lorsque le nombre d'agents de la population utilisant la même stratégie dépasse une certaine valeur de tolérance.

6. Le langage JESS

JESS est l'acronyme de (Java Expert System Shell). Il est défini comme étant : un éditeur de systèmes experts à base de règles et un langage de script à la fois. JESS a été écrit complètement en JAVA et est développé par Ernest Friedman-Hill dans les laboratoires nationaux SANDIA au canada (Hill, 2003). JESS est inspiré du système expert CLIPS (Wygant, 1989). Ainsi, JESS utilise une syntaxe proche du langage LISP (**LIS**t **P**rocessing) pour tout ce qui est manipulation des listes, appels aux méthodes,

insertion des commentaires, etc. Il est également possible de doter des applications JAVA avec un moyen de raisonnement en utilisant JESS. Ce dernier permet une très forte flexibilité en termes de manipulation du code JAVA à l'intérieure d'un script JESS et réciproquement la manipulation des objets JESS (i.e. règles, faits, fonctions … etc.) ou effectuer un raisonnement à l'intérieur d'un code JAVA.

C'est ainsi qu'un système expert sous JESS est composé d'une base de faits, une base de règles et un mécanisme de raisonnement. Une règle JESS est appliquée d'une manière répétitive sur un ensemble de faits dans le but d'avoir, par exemple, de nouveaux faits qui correspondent au contexte d'étude. De plus, les règles sont, spécialement, conçues pour modéliser l'expertise humaine ou les connaissances. Dans la terminologie JESS, on dit une règle déclenchée (*fired* en anglais) ou exécutée pour faire allusion à son application.

Il existe trois manières de représenter les connaissances sous JESS (Menken, 2002), à savoir :

1. *Les règles* : qui sont principalement destinées à représenter la connaissance en se basant sur l'expertise.
2. *Les fonctions* : destinées à représenter les connaissances procédurales. Autrement dit, la représentation des connaissances par le biais des fonctions.
3. *Les modules orientés-objet* : en plus de la représentation des connaissances procédurales, on peut représenter des concepts inhérents à la programmation orienté-objet (POO), à savoir : l'héritage, l'encapsulation, le traitement des messages, l'abstraction et le polymorphisme.

Nous allons expliquer, brièvement, dans ce qui suit les principaux constructeurs du langage JESS.

6.1. Les faits

Dans un système à base de règles, un fait est défini comme étant une portion de connaissance. Ainsi, une collection de faits représente une base de faits. JESS définit trois types de faits, à savoir : les faits ordonnés (*ordered facts*), les faits non-ordonnés (*unordered facts*) et les faits ombres (*shadow facts*).

- *Faits ordonnés*
  Les faits ordonnés sont considérés comme étant des listes où le premier paramètre représente la tête de la liste et la suite des paramètres représente la queue. La tête de la liste consiste en un type pour les autres paramètres (la queue). Dans la figure suivante (figure 2.2), *shoppinglist*, *person* et *father-of* sont des faits ordonnés :

```
1  (shoppinglist eggs milk bread)
2  (person "mohamed sedik" Male 31)
3  (father-of lina sadjed)
```

Figure 2.2 : Un exemple d'un fait ordonné.

- *Les faits non-ordonnés :*

Le concept des faits non-ordonnés dans JESS resemble à celui des objets dans la POO où un objet comporte une instance des attributs déclarés dans la classe. Les attributs dans JESS sont appelés désormais des *slots*.

Dans l'exemple suivant (figure 2.3), *automobile* est un fait non-ordonné comportant les slots : *marque*, *modele*, *couleur* et *annee*.

```
(automobile
    (Marque Citroen)
    (modele berlingo)
    (couleur grise)
    (annee 2011))
```

Figure 2.3 : Un exemple d'un fait non-ordonné.

La déclaration d'un fait non-ordonné dans JESS se fait par le constructeur : *deftemplate* (figure 2.4), sous JESS le fait précédent s'écrit comme suit :

```
(deftemplate automobile
  "Une voiture citroen ..."
  (slot marque
        (type STRING))
  (slot modele
        (type STRING))
  (slot annee
        (type INTEGER))
  (slot couleur
        (type STRING)
        (default white)))
```

Figure 2.4 : Un exemple de la déclaration d'un fait non-ordonné sous JESS.

Pour insérer une instance du fait automobile, nous utilisons la commande JESS *assert* (figure 2.5).

```
(reset)
(assert (automobile
        (marque "citroen")
        (modele "berlingo")
        (annee 2011)))
(facts)
```

Figure 2.5 : Un exemple sur l'insertion d'une instance du fait automobile à la mémoire de travail.

Ainsi, la commande : *facts* (figure 2.6) permet de récupérer l'ensemble des faits qui existent dans la mémoire de travail :

```
f-0   (MAIN::initial-fact)
f-1   (MAIN::automobile (marque "citroen") (modele "berlingo") (annee 2011) (couleur white))
For a total of 2 facts in module MAIN.
```

Figure 2.6 : Récupération des faits de la mémoire de travail.

Un identificateur numérique (*fact-id*) est associé à chaque fait une fois affirmé. Pour récupérer un fait de la base des faits par son identificateur on utilise la commande JESS *retract* (figure 2.7).

```
Jess> (retract (fact-id 1))
  TRUE
  Jess> (facts)
  f-0    (MAIN::initial-fact)
  For a total of 1 facts.
```

Figure 2.7 : Récupération d'un fait par son identifiant.

- *Les faits ombres :*
  Les faits ombres (*shadow facts*) sont à la base des faits non ordonnés qui représentent un pont vers des objets JAVA. L'utilisation des faits ombres permet d'ajouter des objets JAVA dans la mémoire de travail dans le but de les intégrer dans le processus de raisonnement.

6.2. Les règles

Une règle JESS est équivalente à une structure conditionnelle dans la programmation procédurale de type IF... THEN. Ainsi, une règle est divisée en deux parties : gauche ou LHS (Left-Hand Side) et droite ou RHS (Rigth-Hand Side) séparées par => (figure 2.8). La partie gauche d'une règle est employée pour la correspondance des prémisses (pattern matching). En revanche, la partie droite d'une règle représente une liste d'actions à effectuer (i.e. post-condition) si les patterns composant la partie gauche (i.e. pre-condition) sont vérifiés (García-Camino, Noriega, & Rodríguez-Aguilar, 2005).

```
(defrule pre-registration ""
    (AuthorPaperSubmission-Process2
        (agentid ?agID)
        (group ?gr)
        (role ?r)
        (status ?s))
    (rd1
        (RegistrationdeadLine ?rdl)
        (authorRegistrationObligationStarTime ?arost))
    (test (= ?s "waiting"))
    (test (<= ?*currentdate* ?rdl))
    (test (> ?*currentdate* ?arost))
       => (assert (AuthorPaperSubmission-RegistrationObligation
            (agentid ?agID)
            (group ?gr)
            (role ?r)
            (status "preregistred")))
```

Figure 2.8 : Un exemple d'une règle JESS.

6.3. L'agenda

JESS exécute la partie gauche (LHS) de la règle la plus prioritaire dans l'agenda. Cette règle sera retirée par la suite de l'agenda et la suite d'actions de la partie droite (LHS) sera exécutée. Par définition, un agenda JESS comporte la liste des règles activées et qui ne sont pas encore déclenchées. Pour récupérer le contenu de l'agenda on utilise la commande JESS : *agenda* (figure 2.9) :

```
Jess> (agenda)
[Activation: MAIN::duck  f-0 ; time=2 ; salience=0]
For a total of 1 activations.
Jess>
```

Figure 2.9 :L'agenda JESS.

Finalement, la figure suivante (figure2.10) montre l'environnement d'exécution JESS :

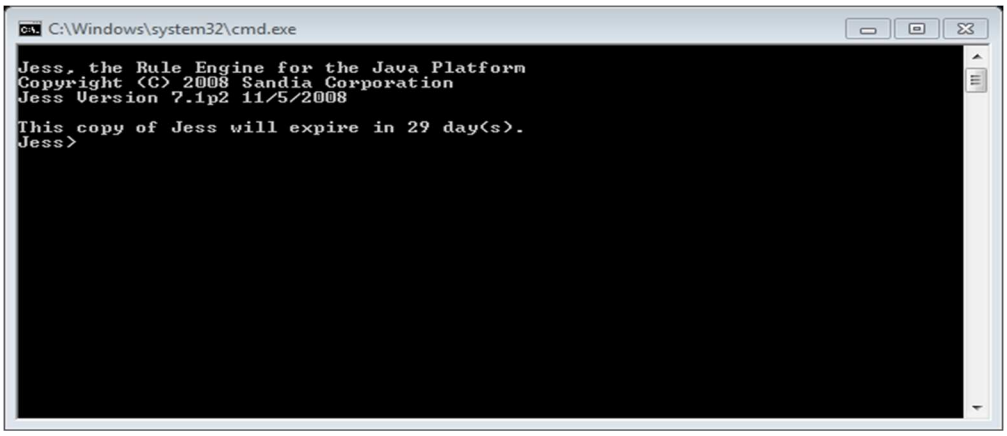


Figure 2.10 : L'exécution de JESS.

7. Monitoring des normes

Le monitoring des normes est divisé grossièrement en deux parties fondamentales : le monitoring par la plateforme agent et le monitoring par l'agent lui-même. Le dernier type de monitoring a été étudié selon plusieurs niveaux d'abstraction à savoir : l'auto-monitoring, le monitoring par une seconde partie et le monitoring par une tierce partie (Criado, Argente, Noriega, & Botti, 2013).

Dans l'auto-monitoring l'agent s'occupe de son propre comportement vis-à-vis de la norme (Hollander & Wu, 2011). Le monitoring des normes par une seconde partie (*second-party observability*) figure principalement dans le cas où les agents, impliqués dans une communication, observent chacun d'eux le comportement de son partenaire d'une manière subjective. Par conséquent, l'agent décidera de sanctionner ou de récompenser son partenaire (Boella & Torre, 2003). Le monitoring par une tierce partie, par opposition au monitoring par une seconde partie, se fait par des agents qui ne sont pas impliqués directement dans une communication (objets ou agent non communicant).

Quant au monitoring par la plateforme agent, des plateformes d'agents normatifs offrent la possibilité d'observer et de renforcer les normes par le moyen des agents spécifiques qui sont à la charge de cette tâche. Cardoso et al. (Cardoso & Oliveira, 2007) proposent une architecture de renforcement des normes dans laquelle le monitoring et le renforcement des normes sont effectués par une entité unique (centralisée). Cette entité s'occupe d'envoyer et de recevoir des messages provenant des agents et décidera par la suite si les agents respectent (ou violent) la norme. Ensuite, l'entité centrale répond par envoyer des notifications de sanction (ou de récompense). Dans cette proposition, il est évident que les performances du système vont être certainement affectées considérablement par la surcharge de communication imposée par le concept de centralisation. Une solution du problème de centralisation a été proposée par (Gaertner, Garcia-Camino, Noriega, Rodriguez-Aguilar, & Vasconcelos, 2007) où une architecture distribuée pour le renforcement des normes a été proposée. À cet effet, nous allons adopter dans le cadre de notre thèse une approche distribuée pour le monitoring des normes en profitant de la répartition organisationnelle incorporée dans le modèle AGR. Plus de clarifications sur cette idée seront détaillées dans le dernier chapitre.

8. Mécanismes de renforcement des normes dans les SMA Normatifs

Dans un SMA normatif, les normes sont imposées dans le but de régulariser et de diriger les comportements des agents. Ainsi, l'utilisation des normes offre un moyen accéléré pour arriver à une cible souhaitée dans un environnement complexe et dynamique (Criado, 2013). Cependant, les agents peuvent décider de ne pas se conformer aux normes si cela leur est bénéfique. À cet effet, le renforcement des normes désigne les mécanismes motivant les agents pour qu'ils puissent

s'accommoder avec le comportement normatif. Dans la littérature, les actes de coercition utilisés pour exprimer une exécution sociale sont exprimés en termes de sanctions (Hollander & Wu, 2011). Une sanction peut être une punition pour avoir désobéi à une norme ou une récompense dans le cas opposé. Ainsi, l'acte de sanction est associé avec un coup qui doit être payé par l'agent sanctionné. Plusieurs types de sanctions ont été définies dans la littérature en se basant sur des points intrinsèques à savoir les émotions, la réputation ou même des sanctions prenant la forme de rupture des relations ou de manque de confiance avec l'agent fautif (Verhagen, 2001)(Hollander & Wu, 2011)(Savarimuthu, 2011). D'après Cardoso et al. (Cardoso & Oliveira, 2009), deux catégories basiques regroupant les types de sanctions possibles : les sanctions directes et les sanctions indirectes.

- *Les sanctions directes* : deux stratégies possibles pour appliquer une sanction directement (figure 2.11), soit la dissuasion pour décourager toute infraction future ou la rétribution visant à indemniser la victime de l'infraction (Cardoso & Oliveira, 2009). D'autres mesures sanctionnatrices ont été ajoutées par Vàzquez-Salceda et al. (Vázquez-Salceda, Aldewereld, & Dignum, 2005) consistant à utiliser des solutions techniques à la situation rencontrée, par exemple : des listes noires, des déclencheurs d'horloge, etc.
- *Les sanctions indirectes* : l'effet de ce type de sanction se propage avec le temps en affectant les émotions et la réputation de l'agent (figure 2.11). Ainsi, le terme réputation prend plusieurs sémantiques selon le contexte dans lequel il est employé. Un mécanisme de réputation consiste à donner un avis positif à l'agent accomplissant la norme et inversement attraper l'agent violeur de la norme par un avis négatif. L'avis attribué à un agent que ce soit positif ou négatif est exprimé réellement par l'assignement d'une valeur de réputation qui s'incrémente ou se décrémente selon la conformité du comportement de l'agent vis-à-vis la norme (Grizard, Vercouter, Stratulat, & Muller, 2006). On peut faire allusion aux infractions à la circulation routière où une note est attribuée au permis de conduire. Cette note sera décrémentée (ou incrémenté) selon le nombre d'infractions commises.

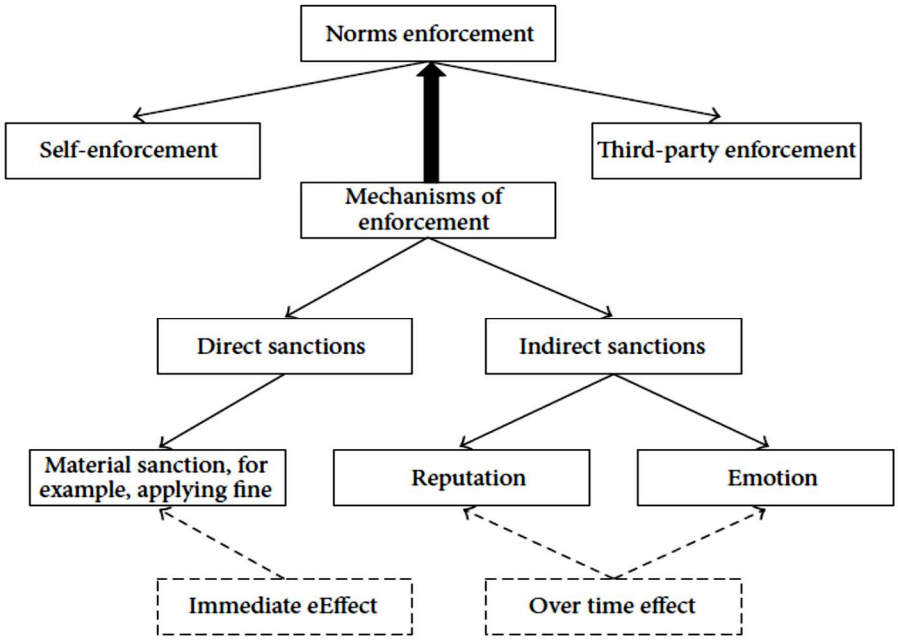


Figure 2.11 : Les types et les mécanismes de renforcement des normes (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014).

Comme le mécanisme de renforcement des normes est vivement lié au mécanisme de monitoring des normes, nous faisons, dans le cadre de cette thèse, la distinction entre monitoring et renforcement. Cela est dû à la nature de la tâche à entreprendre. Pour le cas du monitoring, une entité particulière consiste en l'observation du comportement des agents selon trois manières différentes : l'auto-monitoring, le monitoring par une seconde partie et le monitoring par une tierce partie. Tandis que la tâche de renforcement consiste à appliquer la sanction associée à la norme, pour cela une entité particulière dans le système normatif s'en occupe. Dans la littérature, le renforcement des normes s'effectue de trois manières distinctes (figure 2.11), à savoir, extérieurement, intérieurement, ou par motivation (Young, 2008).

- *L'auto-renforcement* : connu également sous le nom : renforcement dirigé intérieurement (*internally directed enforcement*). Cela se produit lorsqu'un agent se punit lui-même pour avoir violé une norme dans le cas où l'agent a internalisé la norme et est influencé par certaines émotions. Selon (von Scheve, Moldt, Fix, & von Lude, 2005), les émotions représentent un facteur critique qui dirige l'auto-renforcement.
- *Le renforcement par une tierce partie* : connu également par : renforcement dirigé extérieurement (externally directed enforcement). Ainsi, le renforcement par une tierce partie se produit dans le cas où l'agent qui s'occupe du monitoring observe une désobéissance à la norme, il informera en conséquence l'agent renforceur (policier par exemple) pour appliquer la sanction correspondante. Dans les sociétés d'agent, le renforcement extérieur s'applique dans le but de restreindre

et réduire des comportements indésirables dans la population (Caldas & Coelho, 1999).

- *Le renforcement par motivation* : est un type de renforcement social résultant d'un *esprit commun* entre des agents rationnels de la société ou pour des motifs de coordination. Autrement dit, si un comportement particulier (ou une norme) est attendu de tous les agents, alors il n'y a aucun avantage pour un agent à le violer. Ainsi, le seul choix rationnel pour l'agent est d'obéir à la norme (Hollander & Wu, 2011).

9. Les normes comme étant un moyen de contrôle des SMA

Selon (ISO/IEC/IEEE 24765, 2017), le terme contrôle a été intimement lié avec le terme monitoring et est défini de la manière suivante :

> *« In engineering, the monitoring of system output to compare with expected output and taking corrective action when the actual output does not match the expected output. »*

Ainsi, un processus de contrôle prend en entrée un SMA dans n'importe quel état et produit en sortie un système dont son état correspond à la cible attendue. En revanche, le terme *contrôlabilité* (*reachability* en anglais) n'a pas été encore standardisé. Selon (Sontag, 2013), la contrôlabilité est une propriété importante du système contrôlé. Grossièrement, le terme contrôlabilité dénote la capacité de déplacer l'état d'un système dans un moment donné avec toute sa configuration actuelle en utilisant uniquement certaines manipulations potentielles (Ogata, 2009) (Sontag, 2013). Ainsi, la contrôlabilité et le monitoring sont des aspects duels du même problème.

10. Travaux utilisant les normes pour le contrôle des SMA

Dans le but de bien cerner et positionner notre travail, nous allons dans ce qui suit exposer les travaux les plus importants utilisant les normes comme étant un moyen de contrôle des SMA. On conclut par l'élaboration d'un bilan sur le domaine des normes et les SMA normatifs :

10.1. Langages normatifs

- **Dastani et al (Dastani & Torre, 2004)**

Ce travail s'inscrit dans la catégorie des travaux traitant l'aspect normatif pour les agents BDI à l'instar des travaux de (Dignum, Morley, Sonenberg, & Cavedon, 2000) et (Broersen, Dastani, & Torre, 2001). Les auteurs de ces travaux ont intégré les normes de type *obligation* dans l'architecture des agents BDI (Belief, Desire, Intention) et ont considéré les obligations comme un type de connaissance. Dignum et al. (Dignum, Morley, Sonenberg, & Cavedon, 2000) ont proposé de traiter un type particulier de

normes dites : *introspectives*, c'est-à-dire des normes ayant la condition de déclenchement constituée d'obligations ou d'autres éléments de type cognitif (intentions, croyances, désirs) (Stratulat, 2002).

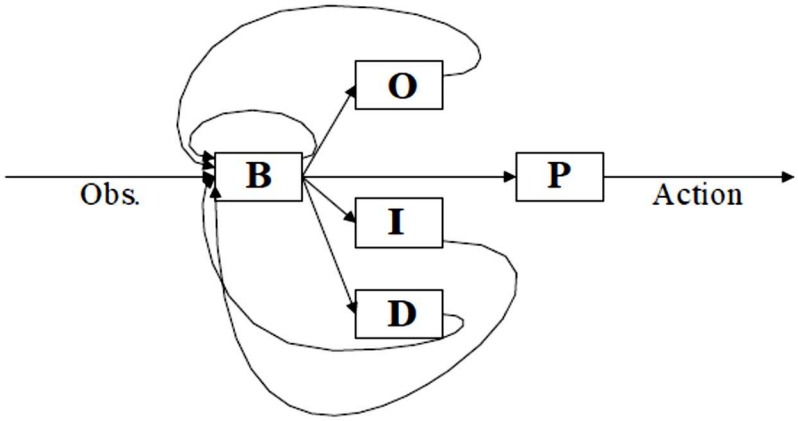


Figure 2.12 :L'architecture BOID (Stratulat, 2002).

Ainsi, Dastani et al. introduisent, dans ce travail, une sémantique opérationnelle d'un langage de programmation abstrait utilisé pour implémenter des agents cognitifs. Ils ont modélisé les attitudes mentales en se basant sur des règles inspirées de l'architecture BOID (Belief, Obligation, Desire, Intention) (Broersen, Dastani, & Torre, 2001). BOID est basé sur des règles de la logique propositionnelle. En observant l'environnement, les agents génèrent un ensemble de buts à partir de leurs désirs, obligations et intentions et ils peuvent également sélectionner des buts, générer des plans et les exécuter en conséquence (figure 2.12). Dans le processus de délibération, les actions de sélection des buts et la génération des plans à exécuter sont combinées dans le langage délibératif de différentes manières. Dans le niveau d'abstraction du langage proposé, la génération des buts et la planification sont considérées comme étant des procédures de résolution de conflits.

- **García-Camino et al (García-Camino, Noriega, & Rodríguez-Aguilar, 2005)**

Dans ce travail, un langage normatif (figure 2.13) a été introduit dans le but d'offrir une forte expressivité aux termes et concepts liés à la normativité des SMA. La grammaire et la syntaxe du langage proposé ont été formalisées par la notation BNF (Backus-Naur Form) et ont été implémentées en utilisant le langage JESS (Friedman-Hill, 2003). Ainsi, dans cette proposition un type particulier de normes a été traité. Il s'agit des normes conditionnées par le temps (i.e. une norme qui est activée pendant un intervalle de temps).

C'est ainsi que le langage proposé est bien dédié à un type de système connu sous le nom *Electronic Institutions* (EI) où le comportement des agents est bien restreint afin d'assurer un environnement sûr pour les interactions et de munir ses composants déontiques de plus de flexibilité et de généralité.

$$
\begin{aligned}
NORM &:= N(\ utter(S,W,I)\ \langle TIME\rangle\ \langle \text{IF } C\rangle\ ) \\
N &:= \text{ OBLIGED} \mid \text{PERMITTED} \mid \\
&\quad \text{FORBIDDEN} \\
I &:= \iota(A,R,A,R,M,T) \\
TIME &:= \text{BEFORE } D \mid \text{AFTER } D \mid \\
&\quad \text{BETWEEN } (D,D) \mid \\
&\quad \text{BEFORE } uttered(S^*,W^*,I^*) \mid \\
&\quad \text{AFTER } uttered(S^*,W^*,I^*) \mid \\
&\quad \text{BETWEEN } (\ uttered(S^*,W^*,I^*), \\
&\qquad\qquad uttered(S^*,W^*,I^*)\ ) \\
C &:= \neg\ (CONDS) \mid CONDS \\
CONDS &:= \langle\neg\rangle COND\ \langle, C\rangle \\
COND &:= V\ OP\ V \mid uttered(S^*,W^*,I^*) \mid \\
&\quad N(\ utter(S^*,W^*,I^*)\ ) \mid predicate \\
V &:= AT \mid F \mid value \\
AT &:= identifier.attribute \mid variable \\
OP &:= \ > \mid < \mid \geq \mid \leq \mid = \\
SANCTION &:= \text{SANCTION } (\ (COMMS) \text{ IF } NP\ (NORM)\ ) \\
NP &:= \text{VIOLATED} \mid \text{COMPLIED} \\
COMMS &:= COMM\langle, COMMS\rangle \\
COMM &:= AT\ =\ F \mid F \\
F &:= identifier(\ <ARGS>\ ) \\
ARGS &:= V\ <,\ V>
\end{aligned}
$$

Figure 2.13 : La notation BNF du langage normatif proposé(García-Camino, Noriega, & Rodríguez-Aguilar, 2005).

Où :

- **S** est l'identificateur de scène. Dans les IE, une institution doit passer par quatre scènes possibles, à savoir : *enregistrement*, *enchère*, *payement* et *livraison*,
- **W** est l'identificateur d'état de l'agent,
- ***i*** : est une particule illocutionnaire comportant :
  - A : identificateur de l'agent (émetteur ou récepteur)
  - R : le rôle de l'agent,
  - M : le contenu du message échangé,
  - T : le temps écoulé
- **D** : la date limite (deadline),
- **S*, W*, I*, A*, R*, M*, T*** : sont des expressions qui peuvent contenir des variables (instanciation) referrant respectivement à : scène, état, illocution, identificateur de l'agent, identificateur de rôle, message et le temps écoulé ;

- ***predicate*** : est une formule du premier ordre dont ses variables sont universellement qualifiées.
- ***utter(s*; w*; i*)*** : est le prédicat qui représente une action qui n'a pas encore été effectuée. Signifie : soumettre une illocution dans l'état w* de la scène s*.
- ***uttered(s*; w*; i*)*** : signifie que l'action demandée (acte illocutoire) a été effectuée.
- Une norme N de type NORM selon la notation BNF peut être une Obligation (OBLIGED), permission (PERMITTED) ou Interdiction (FORBIDDEN).
- **IF** : est un constructeur qui est introduit pour imposer des conditions sur des variables.
- **AT** : dénote comment les valeurs des attributs sont accessibles. Ainsi, *identifier:attribute* dénote que la valeur de *attribute* de l'agent où l'object portant le nom *identifier* a été récupérée.
- **COMMS** : est une fonction qui va être exécutée dans le cas où une norme a été violée ou respectée et par conséquent une sanction ou une récompense doit avoir lieu.

La figure suivante (figure 2.14) montre un exemple d'utilisation du langage proposé pour le cas d'une action soumise à une obligation conditionnée par le temps (deadline). Une fois l'agent gagne l'enchère, il doit passer à la scène de payement pour payer les biens achetés. Le payement doit avoir lieu avant (BEFORE) la clôture de la scène de payement. Si l'agent A jouant le rôle d'un acheteur, a soumis une demande à l'agent C jouant le rôle d'un commissaire-priseur pour un bien au prix P et le crédit de A est supérieur à P, l'agent A est obligé de payer avant que la scène de payement soit clôturée.

$$
\begin{array}{rl}
\text{OBLIGED} & (utter(payment, W, \\
 & \quad inform(A, buyer, B, payee, pay(IT, P))) \\
\text{BEFORE} & uttered(payment, w_5, \\
 & \quad inform(B, payee, all, buyer, \\
 & \quad\quad close())) \\
\text{IF} & uttered(auction, w_2, \\
 & \quad inform(A, auctioneer, all, buyer, \\
 & \quad\quad sold(IT, P, C))), \\
 & A.credit > P
\end{array}
$$

Figure 2.14 : Un exemple d'une obligation avec date limite (García-Camino, Noriega, & Rodríguez-Aguilar, 2005).

Par ailleurs, on peut noter que le langage proposé offre une flexibilité d'expression pour décrire des termes normatifs conditionnés par le temps et également la soumission des normes à des tests conditionnels.

10.2. Représentation des normes

- **Felicíssimo et al (Felicíssimo, Chopinaud, Briot, Seghrouchni, & Lucena, 2008)**

Dans ce travail, une méthodologie nommée DynaCROM (**DYNA**mic **C**ontextual **R**egulation information provision in **O**pen **M**ASs) a été présentée. DynaCROM consiste en l'implémentation dynamique des normes pour des SMA ouverts. Les informations normatives dans DynaCROM ont été classées autour d'un contexte règlementaire composé par : l'organisation, le rôle, l'environnement et l'interaction. Ainsi, les concepts du contexte règlementaire ont été différenciés par les limites de leurs données à savoir :

- Normes de l'environnement : sont appliquées pour tous les agents dans l'environnement règlementé,
- Normes de l'organisation : sont applicables sur tous les agents de l'organisation règlementée,
- Normes du rôle : sont applicables sur tous les agents jouant des rôles,
- Normes d'interaction : sont applicables sur tous les agents impliqués dans une interaction.

Par ailleurs, pour la représentation explicite des normes, DynaCROM utilise une ontologie normative contextuelle. Une ontologie contextuelle, d'après les auteurs, est une ontologie dans laquelle des informations locales à un domaine bien déterminé ont été représentées. En ce qui concerne le renforcement des normes, DynaCROM a été étendue avec SCAAR (Chopinaud, Seghrouchni, & Taillibert, 2006). SCAAR est l'acronyme de : **S**elf-**C**ontrolled **A**utonomous **A**gents gene**R**ator, un mécanisme qui permet de doter des agents par des capacités d'auto-monitoring des normes dans le but d'éviter la violation des normes.

La figure suivante (figure 2.15) montre l'ontologie DynaCROM. Les normes contextuelles dans DynaCROM ont été modélisées en se basant sur une approche Top-Down sur une méta-ontologie comportant la sémantique des normes et un moteur d'inférence de règles pour la composition des normes contextuelles en relation.

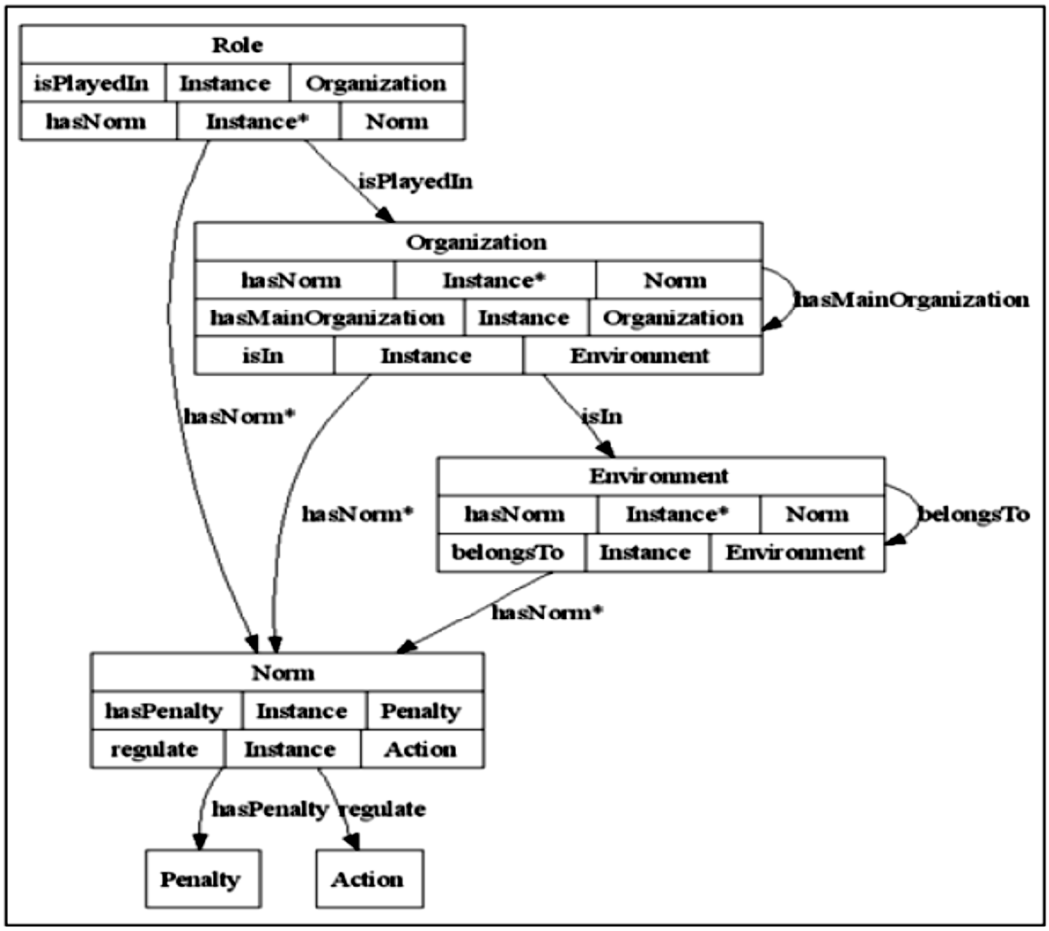


Figure 2.15 : L'ontologie DynaCROM (Felicíssimo, Chopinaud, Briot, Seghrouchni, & Lucena, 2008).

DynaCROM est composée de six concepts en relation, à savoir :

- *Action* : englobe toutes les instances d'une action règlementée.
- *Pénalité* : englobe toutes les instances des amendes qui vont être appliquées dans le cas où la norme n'a pas été respectée.
- *Norme* : englobe toutes les instances des normes d'un contexte règlementaire.
- *Environnement* : englobe toutes les instances d'un environnement règlementé et les normes qui lui sont associées.
- *Organisation* : englobe toutes les instances d'une organisation règlementée.
- *Rôle* : englobe toutes les instances d'un rôle règlementé.

10.3. Modélisation des normes

- **Figueiredo et al. (Figueiredo & Silva , 2011)**

Les auteurs proposent, dans le cadre de ce travail, un langage de modélisation des normes sous le nom : NormML pour : *Norm Modeling Language*. NormML permet la modélisation des principales propriétés et caractéristiques des normes. Ainsi,

NormML étend la version préliminaire présentée dans (Silva, Braga, & Figueiredo, 2011).

Dans NormML, les normes ont été représentées dans la phase de conception. Cela est justifié par le fait que la modélisation des normes est une étape très importante dans la spécification des SMA. Par ailleurs, un autre point soulevé dans NormML est celui de la résolution du conflit produit par les normes contradictoires. Par normes contradictoires on désigne la situation où une action est autorisée (permise ou recommandée) par une norme et interdite au même temps par une autre norme.

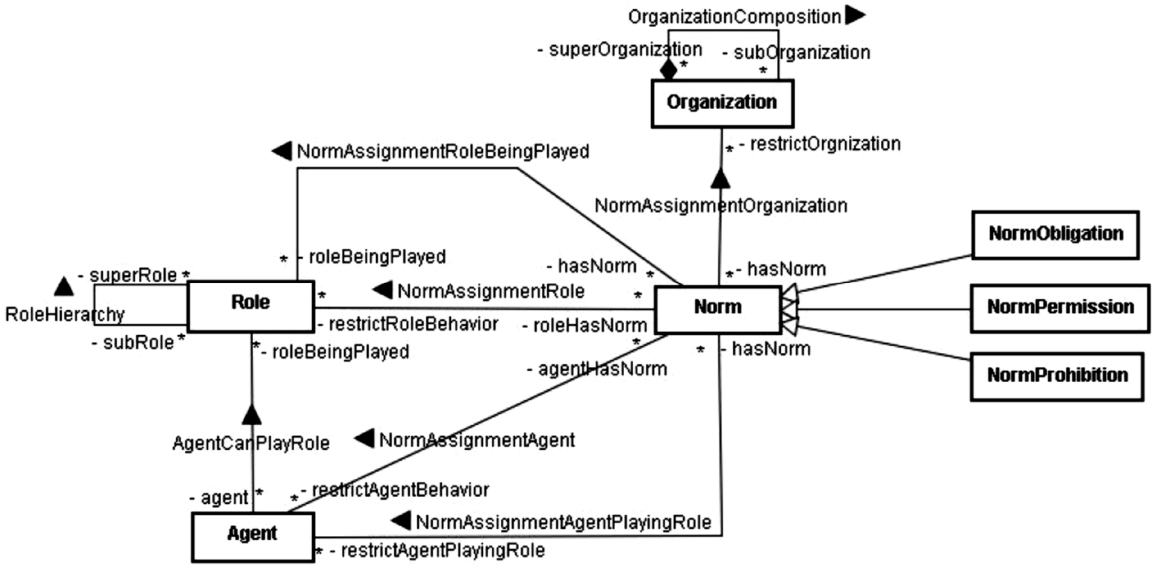


Figure 2.16 : Le méta-modèle de NormML (da Silva Figueiredo, Torres da Silva, & de Oliveira Braga, 2011).

NormML se focalise sur six caractéristiques intrinsèques (figure 2.16) à la norme, à savoir :

- *Le concept déontique* : les concepts déontiques décrivent comment les contraintes comportementales seront imposées en termes de : obligation, interdiction ou permission.
- *Les entités impliquées* : désigne les agents ou les groupes d'agents dont leurs rôles sont mis à des contraintes dérivées dans les normes.
- *Les actions* : représentent l'objet principal d'une norme dans lesquelles des obligations, des permissions ou des interdictions vont être appliquées.
- *Contraintes d'activations* : chaque norme a une période dans laquelle elle est activée (i.e. la durée nécessaire pour l'accomplissement de la norme). Une norme peut être activée (i.e. déclenchée) par un ensemble de contraintes à savoir : les intervalles de temps : *avant*, *après* ou *entre*, l'achèvement de l'état du système cible souhaité, et également l'activation ou la désactivation des autres normes ou l'accomplissement ou la violation des autres normes.

- *Sanctions* : si les contraintes comportementales exprimées dans la norme ont été respectées, l'entité en question sera récompensée sinon elle sera pénalisée. Dans le jargon des SMA normatifs (et également dans NormML), les deux concepts : récompense et pénalité s'intègrent dans la notion de sanction.
- *Le contexte* : désigne le contexte dans lequel la norme sera renforcée (i.e. champ d'application)

10.4. Résolution de conflits

- **Figueiredo et al. (Figueiredo & Silva , 2011)**

Le langage NormML présenté dans la section précédente traite également la résolution de conflits provenant des normes. Ainsi, la résolution de conflits dans NormML a été effectuée selon plusieurs contraintes. Autrement dit : une norme N1 est en conflit avec une norme N2 si les critères suivants sont vérifiés :

- Les deux normes sont définies dans le même contexte.
- Les deux normes sont adressées à la même entité (agent ou rôle).
- Le concept déontique de la norme N1 définit une permission et le concept déontique de la norme N2 définit une interdiction, ou une obligation vis-à-vis une interdiction dans la même période d'activation des deux normes. Le code OCL suivante montre une opération de vérification pour N1 et N2 :

```
Context Set{Norm}::checkDeonticConcept
                                    (n1:Norm,n2:Norm):Boolean
body: if((n1.oclIsTypeOf(NormProhibition))
                        and (n2.oclIsTypeOf(NormObligation)))
then(true)
else(if((n1.oclIsTypeOf(NormProhibition))
                         and (n2.oclIsTypeOf(NormPermission)))
     then(true)
     else(if((n1.oclIsTypeOf(NormObligation))
                         and (n2.oclIsTypeOf(NormPermission)))
          then (true)
          else (false)endif)endif)endif
```

Figure 2.17 : La vérification de conflit des opérateurs déontiques (Figueiredo & Silva , 2011).

- N1 définit un accès en écriture à une ressource (un fichier par exemple) et N2 définit un accès exclusif en lecture (ou mise à jour) en même temps.

- **Ahmed et al (Ahmad, et al., 2011)**

Dans ce travail, les auteurs ont proposé un cadre normatif appelé : OP-RND pour Obligation Prohibition Recommended Neutrality Disliked framework pour résoudre le conflit qui existe entre les buts propres à un agent et les buts décrits dans les normes dans un SMA normatif.

D'après les auteurs, le conflit survient dans le processus de prise de décision pour les buts (personnel ou normatif) qui sont atteints en se basant sur les tâches soumises à des contraintes temporelles. Ainsi, les agents effectuent leurs tâches à partir d'un ensemble de tâches pré-complieés en se basant sur leurs croyances acquises sur les récompenses et les sanctions associées à la tâche sélectionnée. Ils ont défini deux opérateurs déontiques à savoir :

- Une *obligation* qui est imposée par des agents autoritaires dans le système. Une obligation permet à un agent d'être récompensé s'il a effectué l'action associée et d'être sanctionné en revanche.
- Une *interdiction* qui est définie comme étant un ordre dans lequel l'agent doit éviter d'effectuer l'action correspondante et avoir, en conséquence, une récompsence en quittant l'action sous-jacente ou une pénalité s'il l'a effectuée.

Les obligations et les interdictions sont considérées, d'après les auteurs, comme étant des règles normatives (figure 2.18) imposées par des agents autoritaires (législateurs) dans un système normatif.

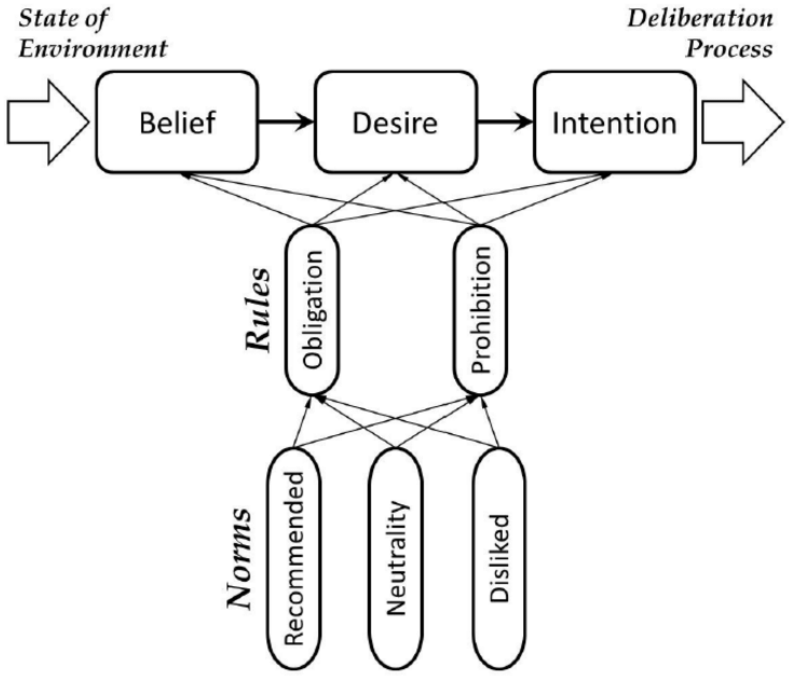


Figure 2.18 : Le cadre normatif OP-RND (Ahmad, et al., 2011).

Pour résoudre le conflit imposé par les contraintes temporelles sur les tâches, deux fonctions ont été introduites à savoir : *sacrifice* et *diligence*. La fonction *sacrifice* permet à un agent de raisonner et d'abandonner n'importe quelle tâche qui a une basse priorité pour faire face à l'accomplissement du but normatif. Cependant, la fonction *diligence* permet à un agent d'accroitre ses efforts pour atteindre le but normatif dans les situations où des contraintes temporelles s'imposent.

10.5. Renforcement des normes

- **Criado et al (Criado, Argente, Noriega, & Botti, 2013)**

Les auteurs de ce travail proposent une architecture de renforcement des normes dédiée pour les SMA ouverts. L'architecture proposée (figure 2.19) porte l'acronyme : MaNEA pour : *Magentix2 Norm-Enforcing Architecture*, et est intégrée particulièrement avec la plateforme Magentix2 (voir le chapitre 1, section n° 9.1). L'objectif de la proposition consiste à doter la plateforme Magentix2 avec une infrastructure capable de contrôler les normes dans les SMA ouverts dans le cas où des scénarios imprévus peuvent se produire. MaNEA permet également la création et la suppression des normes en même temps que l'activation et la désactivation dynamique des instances des normes.

Tel qu'il est expliqué dans le premier chapitre (section n° 9.1), la plateforme Magentix2 permet le développement des SMA ouverts dans lesquels des agents hétérogènes interagissent et s'organisent dans des organisations virtuelles (Foster, Kesselman, & Tuecke, 2001). Par ailleurs, MaNEA permet à Magentix2 d'incorporer un support normatif dans lequel des normes peuvent être créées, instanciées, modifiées ou supprimées. Il est intéressant de différencier entre la création, l'activation et l'instanciation des normes dans le jargon MaNEA. Une norme instanciée implique sa mise en vigueur après son activation. L'activation d'une norme donnée aura lieu une fois l'agent demande d'accomplir un rôle donné. Par conséquent la norme correspondante sera activée. L'activation d'une norme succède sa création.

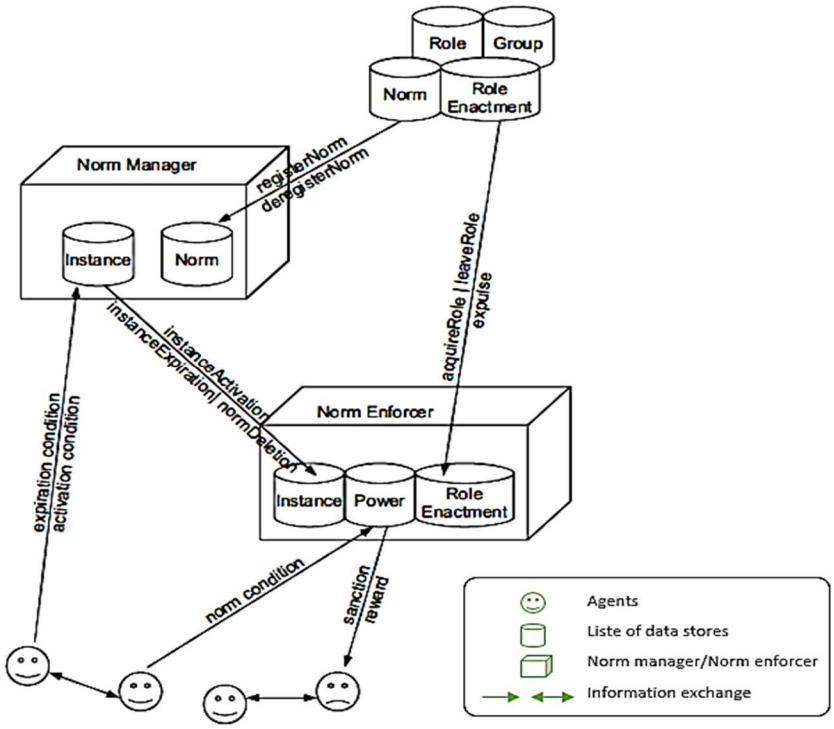


Figure 2.19 : L'architecture MaNEA (Criado, Argente, Noriega, & Botti, 2013).

MaNEA a été conçue suivant une architecture distribuée dans laquelle plusieurs entités ont été développées et dispatchées sur deux couches à savoir :

- La plus haute couche comporte le gestionnaire de normes (*NM : Norm Manager*). Son rôle est d'observer le comportement des agents et détecter l'instance de la norme appropriée.
- La plus basse couche comporte le renforceur (*NE : norm enforcer*). Son rôle consiste au contrôle proprement dit du comportement des agents.

Il est intéressant de noter que MaNEA a été développée à l'aide d'un système de traçage d'évènements dans lequel des évènements qui correspondent aux : création, activation et instanciation de normes ont été captées dynamiquement. Des évènements de type demandes de rôles et de communication ont été également interceptés.

10.6. Cycle de vie de la norme

- **Mahmoud et al (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014)**

Ce travail représente une revue de la littérature spécialisée sur les normes et les SMA normatifs. Ainsi, une classification des normes a été proposée. Les auteurs se sont basés sur la classification de (Coleman, 1998) pour catégoriser les normes dans deux grandes familles à savoir : les normes conventionnelles et les normes essentielles, tel qu'il est expliqué précédemment (section n° 4). Les conventions sont des normes naturelles qui ne nécessitent pas de renforcement. On peut faire allusion aux traditions et coutumes que nous rencontrons habituellement dans la vie quotidienne. C'est ainsi que les normes essentielles sont répertoriées dans trois catégories à savoir : les normes constitutives, les normes régulatrices et les normes procédurales (voir section n° 4).

Un nouveau type de normes régulatrices a été proposé sous le nom : recommandation (figure 2.1). Ainsi, un agent qui exerce une action recommandée sera récompensé tandis qu'il ne sera plus sanctionné autrement. Vu le manque remarquable dans la littérature d'un modèle unifié du cycle de vie de la norme, les auteurs proposent, de plus, un modèle de cycle de vie de la norme (figure 2.20) en se basant sur les travaux traitant le cycle de vie des normes, notamment : (Finnemore & Sikkink, 1998), (Savarimuthu , 2011) et (Hollander & Wu, 2011).

Tel que montré dans la figure suivante (figure 2.20), le cycle de vie de la norme commence par sa création en se basant sur un processus de création qui se déroule indépendamment (*offline design*). La seconde étape consiste en l'émergence de la norme. Cela aura lieu en persuadant les agents à suivre la norme (Finnemore & Sikkink, 1998). L'émergence de la norme est accomplie par une autre opération nommée : opération d'émergence. Cette dernière commence par le renforcement de la norme en utilisant un mécanisme de renforcement donné. Pendant le processus de

renforcement, les agents adoptent la norme par le biais des processus de détection ou de diffusion.

Dans ce travail, Mahmoud et al. ont inspiré les mécanismes de détection et de diffusion de la littérature. Ainsi, quatre mécanismes de détection des normes et trois mécanismes de diffusion ont été adoptés à savoir : l'imitation, l'apprentissage social, le raisonnement à base de cas et le data mining pour la détection et la transmission verticale (*from parents to offspring*), la transmission horizontale (*peers interactions*) et la transmission oblique (*from leader to followers*) pour la diffusion. Ensuite, les deux types de mécanismes assimilent la nouvelle norme dans les groupes de la société. Les agents calculent le coût d'assimilation de la nouvelle norme et décident, en conséquence, d'assimiler ou d'annuler. Après avoir accepté d'assimiler la norme, l'agent intègre, en se basant sur le processus d'internalisation, le raisonnement normatif dans son processus de raisonnement (i.e. raisonnement à base de norme). Finalement, les normes refusées par la société ou par des groupes de la société seront retirées de la structure cognitive des membres de la société.

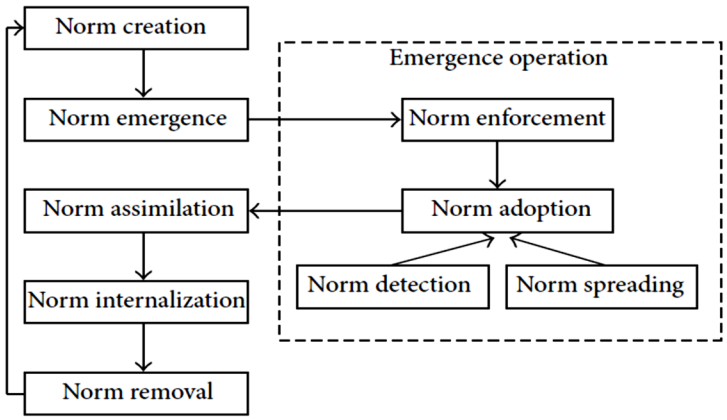


Figure 2.20 : Le modèle de cycle de vie de la norme (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014).

Selon (Hollander & Wu, 2011), les recherches sur l'émergence des normes dans la littérature des SMA normatifs s'articulent sur trois axes principaux : le premier axe porte sur l'utilisation de la théorie des jeux pour expliquer la dynamique de l'émergence de normes. Le deuxième axe examine la relation entre les sanctions et l'émergence des normes. Finalement, le troisième axe tente de comprendre l'impact de la transmission des normes sur leurs émergences.

10.7. Monitoring des normes

- **Alechina et al.(Alechina, Halpern, Kash, & Logan, 2018)**

Dans ce travail, une nouvelle approche décentralisée de monitoring des normes dans les SMA a été introduite. Le monitoring discuté consiste à détecter l'accomplissement

ou la violation des normes par les agents eux-même (i.e. *auto-monitoring* ou encore *decentralised monitoring*) et non par des tiers. La nouveauté de cette approche vient du fait que le SMA ne doit pas en aucun cas supporter des coûts supplémentaires causés par la surcharge de monitoring. Ainsi, les amendes imposées aux agents qui violent les normes ne sont pas considérées inversement à l'idée proposée dans (Fagundes, Ossowski, & Meneguzzi, 2014).

Donc, l'idée clé de cette approche consiste à inciter les agents au sein d'un SMA pour qu'ils puissent surveiller les actions effectuées par les autres agents. Par conséquent, l'agent sera récompensé uniquement dans le cas de détection effective d'une violation de la norme.

10.8. Plateformes normatives

- **José Plácido et al.(da Cunha, Tassio, Marx, & Lucena, 2018)**

Ce travail est, à la base, une extension du travail de (Nunes, Lucena, & Luck, 2012) dans lequel une couche logicielle a été intégrée à la plateforme JADE permettant l'implémentation des agents BDI. L'additif logiciel est déployé sous le nom BDI4JADE. La proposition en elle-même ne permet pas l'adoption de l'architecture BDI connue, mais de proposer un mode de raisonnement étendant l'architecture BDI classique par de nouveaux attributs à savoir : les capacités (*capabilities*), les stratégies (*strategies*) et les évènements (*events*) pour les applications basées JADE.

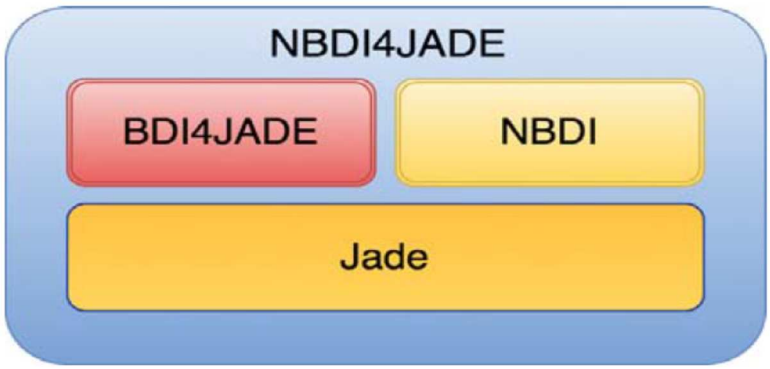


Figure 2.21 : L'architecture NBDI4JADE (da Cunha, Tassio, Marx, & Lucena, 2018).

En revanche, le travail de da Cunha et al. (figure 2.21) augmente l'architecture BDI4JADE par l'ajout des concepts normatifs (Normative BDI4JADE) dans lesquels les normes jouent un rôle crucial dans le processus de raisonnement des agents.

- **Marir et al. (Marir, Silem, Mokhati, Gherbi, & Bali, 2019)**

Dans ce travail, une extension de la plateforme JADE (Bellifemine, Poggi, & Rimassa, 2001) a été proposée sous le nom NorJADE pour : Normative JADE. L'extension proposée consiste à doter les développeurs JADE d'un cadre normatif dans lequel des

mécanismes de renforcement des normes ont été mis à disposition. La plateforme proposée (figure 2.22) profite à la fois des avantages de la POA pour le monitoring des normes et les ontologies pour la représentation des normes.

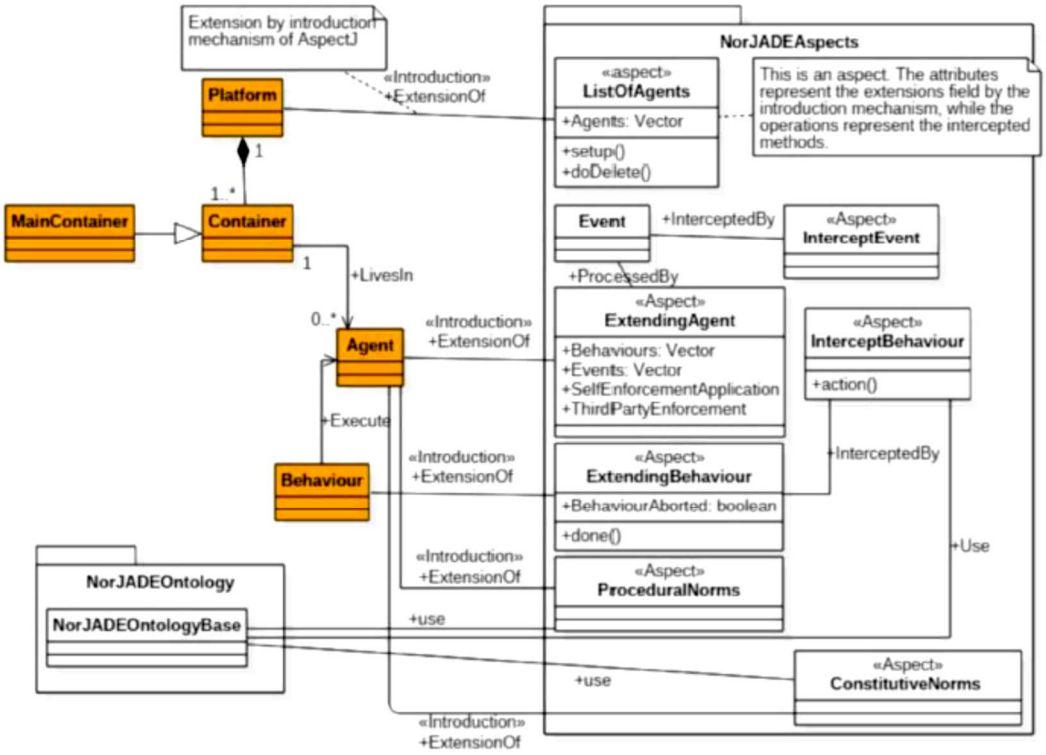


Figure 2.22 : Architecture du framework NorJADE(Marir, Silem, Mokhati, Gherbi, & Bali, 2019)

Le particularité de ce travail réside dans la manière de représenter les normes où maints concepts inhérents ont été traités à savoir :

- La représentation des différents types de normes : constitutives, procédurales et régulatrices.
- La représentation des opérateurs déontiques : obligations, interdiction, permission.
- La représentation des mécanismes de renforcement des normes : l'enrégimentation et le renforcement proprement dit.
- La représentation des concepts relatifs au renforcement des normes à savoir : la sanction et la récompense et, également, les types de renforcement : l'auto-renforcement et le renforcement par une tierce partie. Par contre, le renforcement par une seconde partie n'a pas été traité dans NorJADE.
- La représentation des normes soumises à des contraintes temporelles. (i.e. durée de validation des normes).
- La représentation du type de pénalité imposé sur le comportement contrôlé soit par l'exécution d'une méthode ou l'exécution d'un comportement.

NorJADE étend également JADE par des aspects implémentés sous le langage AspectJ qui permettent à la fois d'intercepter l'exécution d'un comportement et l'occurrence des évènements dans le but d'activer la norme adéquate.

11. Bilan

On trouve dans l'état de l'art précédent une variété de travaux traitant la problématique du contrôle des SMA en utilisant les normes comme un mécanisme de contrôle. Vu l'ampleur des concepts liés au domaine des normes et des SMA normatifs, les recherches réalisées traitent particulièrement quelques notions qui correspondent à un contexte d'étude bien déterminé. Ainsi, ces travaux se répartissent en différents niveaux d'abstraction à savoir, les langages normatifs, le cycle de vie de la norme, le renforcement des normes, la représentation des normes, la résolution de conflits entre les normes, le contexte d'application des normes et les plateformes normatives. Nous avons élaboré une étude comparative (table 2.1) sur les travaux sélectionnés dans la section précédente selon plusieurs critères que nous jugeons pertinents. Les critères choisis englobent les principaux axes de recherche couverts par la communauté scientifique spécialisée dans le domaine des normes. Les critères de transmission et d'émergence ont été exclus, car ils font partie de la théorie sociale, mais ils n'ont pas encore été projetés sur les SMA.

Nous avons mis l'accent premièrement sur le critère de renforcement des normes en termes de sanction ou récompense et la manière de détecter la violation ou l'accomplissement des normes. La modification des normes consiste en la mise à jour des normes pour adapter le comportement du système aux besoins fonctionnels nouvellement rencontrés ou pour des raisons d'interblockage. L'architecture distribuée de renforcement consiste à surpasser les inconvénients des architectures centralisées. Les actions effectuées désignent la possibilité de surveiller le cycle de vie des agents. L'échange de message désigne la manière avec laquelle les entités impliquées dans le contrôle se communiquent entre elles. Cela affecte certainement les performances du système et imposera une surcharge de communication additive. Le critère de représentation des normes désigne la possibilité d'utiliser un formalisme donné (logique déontique, systèmes à base de règles, ontologies etc.) pour modéliser les normes. Les concepts déontiques désignent quel type de modalité a été utilisé et également la possibilité d'utiliser de nouvelles modalités (la recommendadtion à titre d'exemple). Le contexte signifie l'aspect couvert par l'utilisation des normes à savoir l'interaction, l'environnement ou l'organisation. Finalement, un critère très important traitant la résolution de conflit entre les normes.

Critères / Proposition	Renforcement des normes		Modification des Normes	Architecture distribuée de renforcement	Actions effectuées	Cycle de vie de la norme	Échange de messages	Représentation des normes	Concepts déontiques				Contexte			Résolution de conflits
	Sanction	Récompense							Permission	Obligation	Interdiction	Recommandation	Interaction	Environnement	Organisation	
Dastani et al.					√			√		√						√
García-Camino et al.	√	√			√			√	√	√	√				√	
Felicíssimo et al.	√	√			√			√	√	√	√		√	√	√	
Figueiredo et al.	√	√			√	√		√	√	√	√			√	√	√
Ahmed et al.	√	√						√		√	√	√				√
Criado et al.	√	√		√	√	√	√	√	√	√	√			√	√	
Mahmoud et al.	√	√				√						√				
Alechina et al.	√	√		√	√			√								
José Plácido et al.	√	√			√	√		√	√	√	√			√		
Marir et al.	√	√			√			√	√	√	√	√				

Table 2.1 : Receuil des approches de contrôle des SMA.

En analysant la table précédente, on constate bien qu'il y a un consensus entre les travaux présentés pour les critères de *renforcement des normes*, *les concepts déontiques* utilisés, *la représentation des normes* et le monitoring de cycle de vie des agents en termes *d'actions effectuées*. Cependant, uniquement deux travaux traitant la *distribution de renforcement des normes*, Criado et al. (Criado, Argente, Noriega, & Botti, 2013) et Alechina et al. (Alechina, Halpern, Kash, & Logan, 2018). Cela montre à la fois que la majorité des travaux se focalisent sur une architecture centralisée et l'importance de la manière de renforcer les normes en utilisant une architecture distribuée. Ainsi, le critère de *résolution de conflit* a été abordé par trois travaux, Dastani et al. (Dastani & Torre, 2004) Figueiredo et al. (Figueiredo & Silva , 2011) et Ahmed et al. (Ahmad, 2012). En fait, nous pensons que la résolution de conflit nécessite l'intervention des approches de spécification et de vérification formelles dans le but de doter les normes avec une sémantique encore plus claire et généralisée. En revanche, *la modification des norme*s souffre encore du manque d'études et représente en conséquence un domaine vierge qui nécessite d'être exploré. Ensuite, le critère de *l'échange de messages* a été abandonné par tous les travaux à l'exception de Criado et al. (Criado, Argente, Noriega, & Botti, 2013), ce qui justifie que les approches proposées autour des normes ont abandonné la manière classique de communication entre les agents impliqués dans le processus de contrôle. Ainsi, une surcharge de communication sera ajoutée par les entités de contrôle de plus la surcharge ordinaire imposée par les agents du système. Une nouvelle manière de s'occuper de ce problème est de profiter des techniques du paradigme orienté aspect qui est effectivement concrétisé dans le travail de Marir et al. (Marir, Silem, Mokhati, Gherbi, & Bali, 2019). De même, un symptôme négatif figure dans le traitement des aspects normatifs est l'omission du *cycle de vie* de la norme. Nous pensons qu'il est irrationnel de traiter la norme en négligeant le traitement de l'ensemble des évènements suivants : création, activation, instanciation, la mise en vigueur et la désactivation. Seul le travail de Criado et al. (Criado, Argente, Noriega, & Botti, 2013) qui a exploré le cycle de vie entier de la norme avec plus d'attention. En revanche, les autres travaux notamment Figueiredo et al. (Figueiredo & Silva , 2011) et da Cunha et al. (da Cunha, Tassio, Marx, & Lucena, 2018) étudient quelques aspects bien déterminés dans le cycle de vie de la norme à savoir la création et l'expiration. Ainsi, Mahmoud et al. (Mahmoud, Ahmad, Yusoff, & Mustapha, 2014) ont enrichi le cycle de vie de la norme par de nouvelles notions à savoir l'émergence et la transmission. Finalement, le critère de l'interaction comme étant un contexte normatif n'a pas été exploré à l'exception du travail de Felicissimo et al. (Felicíssimo, Chopinaud, Briot, Seghrouchni, & Lucena, 2008). Nous confirmons qu'un cadre normatif pour la gestion de la communication dans les SMA serait une proposition à la stature importante vu le caractère émergent qui remonte en raison des communications imprésivibles. Il est encore évident que les travaux présentés ne prennent pas en considération les spécificités des SMA basés sur des modèles

organisationnels. Autrement dit, la manière dont les SMA ouverts sont implémentés n'est plus spécifiée.

En se basant sur le recueil élaboré, les chercheurs ont été vivement invités à couvrir les aspects qui n'ont pas encore été abordés dans le but de garder la continuité des recherches dans le domaine sous-jacent et proposer de nouvelles approches permettant de surmonter les lacunes rencontrées .

12. Conclusion

À travers ce chapitre, nous avons illustré l'apport des normes dans les SMA comme étant un mécanisme influençant. Ainsi, l'utilisation des normes serait indispensable dans notre contexte de recherche vu la variété des notions et des domaines d'études qui l'entourent. Dans ce chapitre, nous avons introduit les normes à travers plusieurs concepts clés à savoir : la définition de la norme, la typologie et la représentation des normes, les mécanismes de renforcement des normes et le cycle de vie de la norme. Un état de l'art regroupant les travaux les plus pertinents a été présenté, ensuite une étude comparative combinant les différentes propositions selon un ensemble de critères a été élaborée.

Dans le chapitre suivant, nous allons quitter un peu les SMA vers un autre axe de recherche traitant le monitoring comme étant un moyen d'observation et d'analyse des systèmes. Ensuite, nous allons étudier les méthodes et les techniques de monitoring discutées sur les SMA. Par ailleurs, un état de l'art regroupant également les travaux pertinents sera élaboré et contesté en mettant l'accent sur plusieurs critères et techniques.

Chapitre 3

MONITORING DES SYSTÈMES MULTI-AGENTS OUVERTS

Sommaire

1. Introduction

Le monitoring consiste en l'observation du comportement des agents par rapport à celui décrit dans la norme. À cet effet, la majorité des approches de contrôle dédiées aux SMA, dans la littérature, ont réduit la phase de monitoring à une étape secondaire dans le processus de contrôle, c'est le cas à titre d'exemple de (Criado, Argente, Noriega, & Botti, 2013) où la notion de monitoring s'attache uniquement à l'aspect normatif.

Nous avons pensé à généraliser le processus de monitoring et le détacher de tout contexte spécifique. Pour cela, nous allons étudier le monitoring dans le sens général du terme où un SMA soumis au monitoring sera surveillé complètement indépendamment du contexte normatif. Autrement dit, nous voulons mettre plus d'attention sur les caractéristiques inhérentes aux SMA telles que la communication, le cycle de vie des agents, le mouvement des agents au sein d'un SMA ouvert, etc. Ainsi, un monitoring orienté-agent serait avantageux vu l'insuffisance remarquable des travaux traitant ce sujet. Un moniteur qui prend en charge les spécificités d'un SMA offre certainement un moyen efficace pour un processus de contrôle.

Nous introduisons dans ce chapitre des concepts de base sur le monitoring, le profilage et l'instrumentation en mettant au clair la différence entre le monitoring et le profilage. Ensuite, nous présentons le langage AspectJ comme étant, à la fois, un langage de Programmation Orienté Aspects (POA) et une technique d'instrumentation prometteuse. Finalement, nous dressons une étude comparative entre quelques propositions de monitoring dans le domaine des SMA suivi d'un bilan récapitulatif sur un ensemble de critères dont le but est de traiter les lacunes détectées dans ces travaux.

2. Concepts et définitions

2.1. Monitoring

Le terme moniteur (en anglais : *monitor*) a été défini dans le glossaire des standards IEEE software engineering – vocabulary (ISO/IEC/IEEE 24765, 2017), comme :

> *« A software tool or hardware device that operates concurrently with a system or component and supervises, records, analyzes, or verifies the operation of the system or component ».*

Dans cette définition, la distinction a été faite entre un moniteur logiciel et un moniteur matériel. Ainsi, un moniteur peut être un outil logiciel ou un périphérique matériel ou une combinaison entre les deux (Waller, 2014). Le moniteur consiste à analyser et vérifier les aspects opérationnels du Système Soumis au Monitoring (SSM, en anglais SUM : System Under Monitoring) qui peut être aussi un logiciel ou matériel.

Nous allons nous concentrer dans ce chapitre sur le monitoring comme étant un outil logiciel dédié particulièrement à la surveillance des autres logiciels.

A cet effet, un moniteur logiciel est défini, selon (ISO/IEC/IEEE 24765, 2017), par :

> *« A software tool that executes concurrently with another program and provides detailed information about the execution of the other program ».*

Cette définition apporte un concept très important exprimé par le terme *concurrently* (en français, parallèlement). Ceci dit, le moniteur s'exécute toujours conjointement avec le SSM. Ainsi, cette définition s'attache amplement avec la définition de l'analyse dynamique où l'analyse s'effectue sur les données collectées pendant l'exécution du SSM (Plattner & Nievergelt, 1981)(Ball, 1999)(Waller, 2014). Tandis que, l'analyse statique consiste à raisonner sur un programme sans l'exécuter. L'analyse statique est utilisée principalement pour *prédire* les performances du système, par contre l'analyse dynamique est utilisée pour *mesurer* les performances du système (Ernst, 2003).

Selon (Dufour, et al., 2004), deux manières possibles pour analyser un système dynamiquement (analyse dynamique) à savoir : *en ligne* ou *hors ligne*. L'analyse dynamique *en ligne* consiste à analyser le système au cours de son exécution. Par conséquent, cela provoque une dégradation dans les performances du système en termes des ressources exploitées. Tandis que dans l'analyse dynamique *hors ligne* le système est évalué après son exécution à l'aide des traces d'exécution. Cela n'affecte plus les performances du système. Ainsi, une quantité énorme de données sera récoltée et est proportionnelle au temps d'exécution.

Par ailleurs, la majorité des travaux de recherche exploite le terme profilage (en anglais : *profiling*) de la même manière pour exprimer le monitoring (Viswanathan & Liang, 2000)(Sabetta & Koziolek, 2008). Néanmoins, Woodside et al. (Woodside, Franks, & Petriu, 2007) à l'instar de Waller Jan (Waller, 2014) a mis l'accent sur les points de différence entre les deux techniques. Le profilage consiste à récupérer des données sur l'exécution du système pendant la phase de développement du système en question. Parfois, le terme profilage est utilisé pour faire allusion à l'étape de test du système (Woodside, Franks, & Petriu, 2007). Le monitoring s'effectue principalement pendant l'exécution du système. Autrement dit, après le déploiement du système. Ainsi, la technique utilisée pour récolter les données fait une distinction intrinsèque entre le monitoring et le profilage. Ce dernier utilise la technique de *l'échantillonnage* qui consiste à exécuter le système pendant des intervalles de temps variés. Le monitoring, en revanche, utilise la technique des *traces d'exécution* qui consiste à traquer les évènements provenant du système depuis le début jusqu'à la fin d'exécution. Par conséquent, la surcharge d'exécution (*overhead*) causée par le monitoring sera minimale, particulièrement pour le cas de l'analyse dynamique hors ligne, comparée avec la surcharge imposée par le profilage qui est souvent

considérable. En ce qui concerne la nature de données récoltées pour les deux techniques, le profilage rassemble des données détaillées et orienter pour un critère de test bien déterminé par contre le monitoring regroupe des données spécifiques à un contexte particulier. Par exemple, dans le cadre de notre thèse, le monitoring consiste à récupérer des données spécifiques au domaine des SMA (cycle de vie des agents, actions effectuées par un agent, les échanges de messages entre les agents, etc.). Ainsi, nous adoptons le terme monitoring à la place du terme profilage. En fait, les caractéristiques de la technique de monitoring montrées précédemment nous mènent à l'adapter dans notre thèse. Plusieurs arguments seront élucidés dans le chapitre suivant justifiant notre choix.

Un récapitulatif (table 3.1) des concepts discutés précédemment a été présenté dans la table suivante :

	Profilage	Monitoring
Temps d'utilisation	Développement	Exécution
Charge de travail (*workload*)	Générée	Réele
Données récoltées	Détaillées	Spécifiques
Surcharge d'exécution (*overhead*)	Élevé	Faible

Table 3.1 : La différence entre profilage et monitoring (Waller, 2014).

2.2. Instrumentation

L'instrumentation est la technique de monitoring permettant la collecte des données sur les SSM. Par définition, le terme instrumentation consiste à ajouter des fragments de code (*sondes,* en Anglais *probes*) à un programme de façon à ajouter la génération des résultats d'analyse à son comportement initial, au cours d'exécution (Binder, Hulaas, & Moret , 2007)(Pearce, Webster, Berry, & Kelly, 2007).

Deux manières possibles sont utilisées pour instrumenter le code source, l'instrumentation statique et l'instrumentation dynamique. L'instrumentation statique se fait au niveau du code source et par conséquent le maintien du code source serait indispensable (Delahaye, 2007). Tandis que l'instrumentation dynamique s'effectue durant la phase d'exécution du système que ce soit pendant le chargement des classes (*bytecode instrumentation*) ou par le biais de la machine virtuelle JAVA (JVM) à travers des crochets ou hameçons (*hooks*) permettant à un utilisateur d'un logiciel de lui demander des fonctions supplémentaires à des moments déterminés.

Une nouvelle approche d'instrumentation (figure 3.1) a été figurée pendant ces deux dernières décennies permettant d'en tirer bénéfices des techniques de la POA. On parle, donc, de la technique de Monitoring Orienté Aspect (MOA) qui se base principalement sur l'Instrumentation Orientée Aspect (IOA). En fait, l'avènement du

paradigme orienté aspect était essentiellement dans le but d'optimiser et de nettoyer le code source d'une application donnée en séparant les préoccupations fonctionnelles de celles transversales. Autrement dit, un code qui n'appartient plus aux exigences fonctionnelles fixées préalablement (comme le code IHM) sera séparé dans des modules spéciaux appelés *Aspects*. Le code des Aspects sera réintégré par la suite dans le code fonctionnel de l'application à travers des mécanismes bien orientés. Le langage le plus dominant du paradigme orienté aspect est AspectJ. Nous allons mettre plus de détails autour de AspectJ dans la section suivante afin de bien clarifier comment s'effectue l'instrumentation orientée aspect.

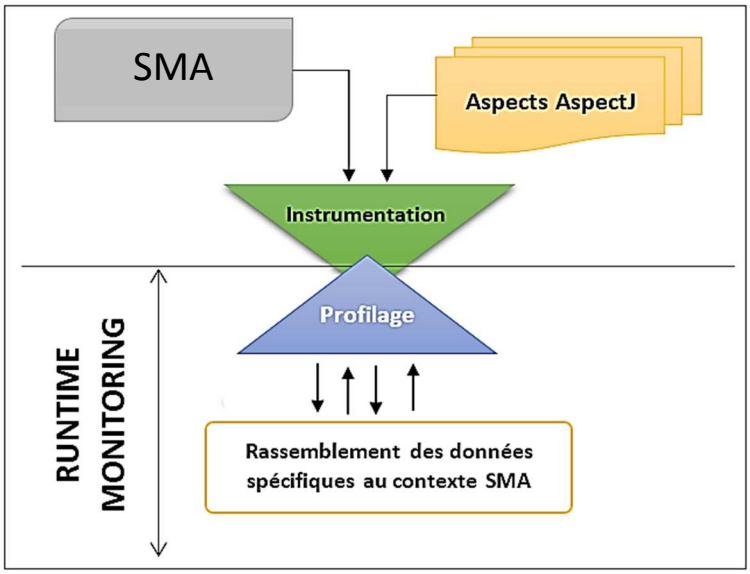


Figure 3.1 : La démarche de monitoring avec AspectJ.

3. Le langage AspectJ

AspectJ (Kiczales, et al., 2001) représente l'extension orientée-aspect du langage JAVA. Initialement, AspectJ a été proposé dans le but d'offrir un moyen pratique pour la séparation de préoccupations transversales de celles fonctionnelles. Ainsi, maints travaux dans la littérature ont utilisé AspectJ à des fins de profilage, nous citons entre autres : (Khaled, Noble, & Biddle, 2003)(Richters & Gogolla, 2003), (Chen & Roşu, 2005), (Avgustinov, et al., 2006), (Bodden & Havelund , 2008) et (Nusayr & Cook, 2009).

Par rapport à la POO, le paradigme aspect a introduit de nouveaux concepts à savoir :

- *Les points de jonction* :
  Un point de jonction représente n'importe quel point d'exécution dans un programme. Ainsi, un point de jonction est la base de l'instrumentation des programmes en utilisant AspectJ (Waller, 2014).
  La table ci-après (table 3.2) montre les points de jonction proposés par AspectJ :

Point de jonction	Description
Method call	Quand une méthode est appelée
Method execution	Quand le corps d'une méthode est exécuté
Constructor call	Quand un constructeur est appelé
Constructor execution	Quand le corps d'un constructeur est exécuté
Static initializer execution	Quand l'initialisation statique d'une classe est exécutée
Object pre-initialization	Avant l'initialisation de l'objet
Object initialization	Quand l'initialisation d'un objet est exécutée
Field reference	Quand un attribut non-constant d'une classe est référencé
Field set	Quand un attribut d'une classe est modifié
Handler execution	Quand un traitement d'une exception est exécuté
Advice execution	Quand le code d'un advice est exécuté

Table 3.2 : Les types de points de jonction AspectJ (Laddad, 2003).

- *Les coupes* :
  Une coupe regroupe plusieurs points de jonction partageant une caractéristique donnée. Par exemple, des méthodes commençant par le mot *get* ou toutes les méthodes de la même classe.
- *Les greffons* (*code advice*) :
  Dans AspectJ, les greffons sont utilisés conjointement avec les coupes. Bien que les coupes capturent les points de jonction et ne font rien d'autre, un greffon est un bloc d'instructions associé à une coupe. En général, les greffons peuvent être définis pour être exécutés *avant*, *après* ou *autour* des points de jonction spécifiés.
- *Aspect* :
  Un aspect est une entité logicielle qui capture une fonctionnalité transversale à une application (Pawlak, Retaillé, & Seinturier, 2004). Ainsi, tous les ingrédients nécessaires pour la définition d'un aspect sont définis à l'intérieur d'un aspect à savoir : les coupes, les points de jonction, les greffons, le mécanisme d'introduction. Techniquement parlant, un aspect resemble à une classe et profite certainement des notions apportées par la POO telles que l'héritage, l'implémentation des interfaces, la visibilité des attributs et des méthodes déclarées dans un aspect est maintenu en utilisant les constructeurs : *private*, *protected* et *public*.
- *Le filtrage* :
  AspectJ propose des mots-clés qui sont utilisés à des fins de filtrage. À titre d'exemple si on veut sélectionner des points de jonction appartenant à une classe particulière en excluant les autres classes (mots-clés : *within*) ou encore faire une sélection sur les points de jonction appartenant dans une méthode (mots-clés *withincode*), etc. Ainsi, il est possible de faire un filtrage basé sur le flot de contrôle en utilisant le mot-clé *cFlow* qui prend en arguments une coupe. Il identifie tous les points de jonction situes entre le moment où l'application passe

par un des points de jonction de la coupe et le moment ou l'application sort de ce point de jonction.

- *Le mécanisme d'introduction* :
  AspectJ offre la possibilité d'introduire de nouveaux membres (attributs ou méthodes) dans des classes ou de changer la relation d'héritage entre les classes.
- *Tissage* (*weaving*) :
  Le tissage (figure 3.2) est le processus qui prend en entrée un ensemble d'aspects et une application de base et fournit en sortie une application dont le comportement et la structure sont étendus par les aspects (Pawlak, Retaillé, & Seinturier, 2004).

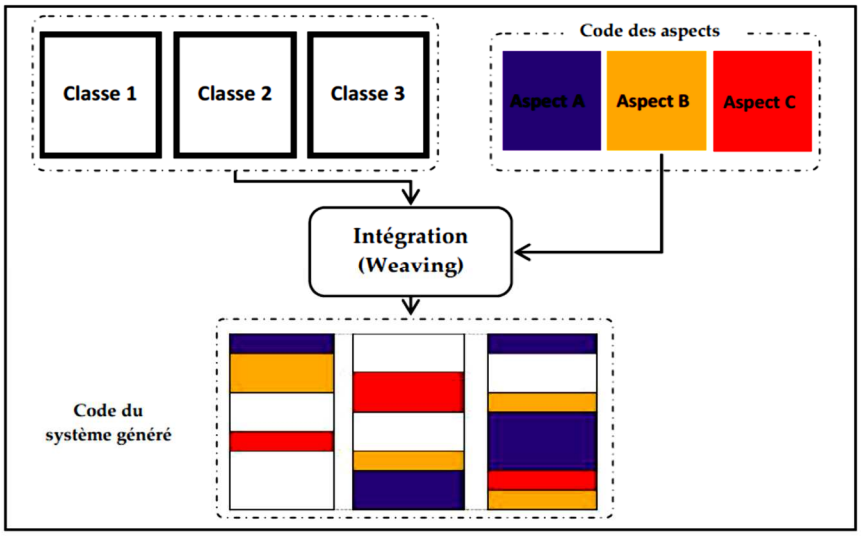


Figure 3.2 : Le tissage des aspects.

Actuellement, AspectJ supporte deux types de tissage : *compile-time weaving* qui s'effectue durant la compilation et *load-time weaving* qui s'effectue durant l'exécution (*run-time*). Le *compile-time weaving* nécessite l'intervention additionnelle du compilateur AJC (AspectJ Compiler) pour compiler le code des aspects. En revanche, dans le *load-time weaving*, une configuration XML est requise pour spécifier les aspects utilisés et leurs paramètres.

4. Instrumentation avec AspectJ

Nous allons montrer à travers un exemple comment s'effectue l'instrumentation par AspectJ. La figure suivante (figure 3.3) montre un fragment de code AspectJ permettant le calcul de temps d'exécution de chaque point de jonction (*méthode*).

L'aspect *profilingAspect* (figure 3.3, ligne 1) définit la coupe *publicOperation* (figure 3.3, ligne2) qui comporte le point de jonction (execution (public * *.*(..)) permettant d'intercepter n'importe quelle *exécution* de méthode dont la visibilité est *publique* qui se situe dans n'importe quel *package* d'où les arguments de cette méthode n'ont pas spécifiés. Cela est indiqué par les deux points de suspension (..) (figure 3.3, ligne 3).

```
1  public aspect ProfilingAspect {
2
3      pointcut publicOperation () : execution (public * *.*(..));
4      Object around() : publicOperation(){
5          long startTime = System.nanoTime();
6          Object r = proceed();
7          long endTime = System.nanoTime();
8
9          System.out.println("Ce point de jonction : "+thisJoinPointStaticPart.getSignature()
10                             + " a pris "+ (endTime-startTime) + " nanos secondes" );
11         return r;
12     }
13 }
```

Figure 3.3 : Un exemple d'un profileur AspectJ simple.

Ensuite, la coupe a été manipulée par le greffon *around* (figure 3.3, ligne 4) qui consiste à mettre un contrôle (avant, pendant et après) autour de la coupe. Autrement dit, le code du point de jonction intercepté est exécuté à l'intérieur du greffon *around* par le biais de la méthode *proceed()* (figure 3.3, ligne 6). Pour cela, des variables entières ont été maintenues permettant la récupération du temps en cours avant et après l'exécution du point de jonction. Ensuite, on peut savoir le temps consommé par l'exécution du point de jonction (figure 3.3, ligne 9).

5. Profilage avec AspectJ

AspectJ a démontré efficacement une intégration flexible avec les applications JAVA. Ainsi, les constructions du langage AspectJ (les greffons, les coupes, les points de jonction, etc.) apportent une surcharge (*overhead*) non significative, voire nulle, dans le cas où ces constructions ont été utilisées d'une manière basique dans des applications JAVA typiques (Dufour, et al., 2004), (Pearce, Webster, Berry, & Kelly, 2007) et (Chebout, Mokhati, & Badri, 2015). Selon (Dufour, et al., 2004) : "*As a percentage of the total execution time, the cost of advice is therefore insignificant in such applications*". Cependant, nous avons démontré dans (Chebout, Mokhati, & Badri, 2015) que les greffons de type *around* et *after returning* apportent une surcharge de communication significative due à l'encapsulation et au dépaquetage des points de jonction à l'intérieur de l'aspect.

En d'autres termes, nous avons essayé de comprendre l'impact des constructions AspectJ sur des caractéristiques inhérentes aux agents. Ensuite, nous avons sélectionné la communication comme critère d'évaluation entre plusieurs versions d'une même étude de cas implémentés à la fois avec JAVA et AspectJ. Le choix de la communication a été motivé par le fait que les agents doivent passer là-dessus dans n'importe quel contexte d'exécution à savoir : la négociation, la collaboration, la coordination …etc. À cet effet, la communication représente une brique de base voire le noyau de tout type d'interaction entre agents.

Ensuite, nous avons sélectionné quelques primitives de communication utilisées par les agents telles que : *sendMessage*, *receiveMessage*, *purgeMailBoxe*, etc. dans le but

d'implémenter la version AspectJ correspondante. La version AspectJ implémentée utilise, particulièrement, des greffons de type *arround*, *afterReturning (AR)* et *afterReturning* combiné avec *cFlow (AR & cF)*. L'évaluation a été faite sur une étude de cas lancée pour une durée de 10 minutes. La figure suivante (figure 3.4) montre l'effet de chaque greffon en termes de temps moyen apporté par ces greffons au temps d'exécution par rapport à la version originale. Ainsi, les greffons de type *arround* apportent une moyenne de temps de 0.778 minute de plus et les greffons de type *around* et *around* combiné avec *cFlow* apportent respectivement aux moyennes 0.784 et 0.783 minute de temps de plus.

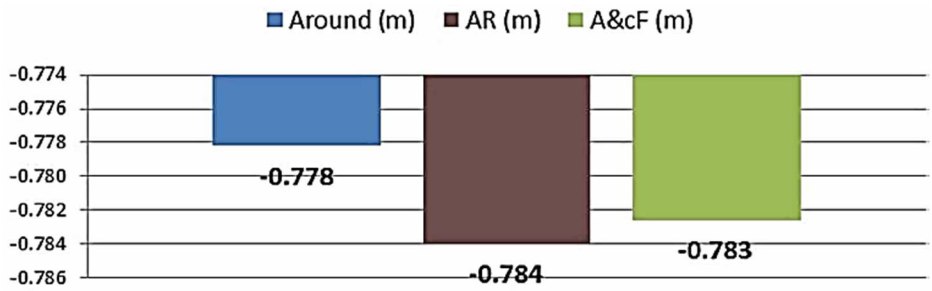


Figure 3.4 : L'impact de temps moyen de chaque greffon par rapport au nombre total d'agents.(Chebout, Mokhati, & Badri, 2015).

Le langage AspectJ a démontré son importance comme étant une technique prometteuse pour répondre aux problèmes de la séparation des préoccupations transversales de celles fonctionnelles et, également, offrir un moyen de profilage de qualité vis-à-vis des outils d'instrumentation classiques. Ainsi, AspectJ a prouvé suffisamment son expressivité (Pearce, Webster, Berry, & Kelly, 2007) pour la majorité des systèmes qui existent dans la littérature (*legacy systems*). Par conséquent, on peut migrer un système implémenté avec JAVA vers sa version AspectJ. Cependant, quelques limites ont contourné le langage AspectJ à savoir :

- Temps de tissage (*Load-time weaving*) tel qu'indiqué dans (Pearce, Webster, Berry, & Kelly, 2007),
- Le problème de l'accès concurrent aux structures de données partageables entre AspectJ et l'application profilée (i.e. *HashMap vs concurrentHashMap*),
- Le niveau de contrôle autorisé par AspectJ est assez abstrait et il ne permet que l'identification des appels ou l'exécution des méthodes ou constructions. Cependant, AspectJ ne permet pas un contrôle assez concret capable d'identifier des structures de contrôle conditionnelles ou répétitives (c.-à-d. boucles, instructions *if … Then*, *switch … case*,etc.).

6. Monitoring orienté-agent

Dans cette section, nous allons exposer quelques travaux traitant du monitoring dans le contexte des SMA. Ainsi, le monitoring dédié aux applications basées agent consiste

en la récolte des données de nature spécifique (nombre de messages échangés, l'impact des messages échangés sur le comportement des agents, les actions effectuées par un agent donné), à la différence des moniteurs classiques où l'objectif est l'évaluation des performances du SSM.

On peut distinguer deux catégories de monitoring : le monitoring par les plateformes agents et le monitoring par une tierce partie. Dans la première catégorie on trouve, à titre d'exemple, les plateformes : ZEUS (Nwana, Ndumu, Lee, & Collis, 1999), JADE (Bellifemine, Poggi, & Rimassa, 2001) et COUGAAR (Helsinger, Thome, & Wright , 2004). La plateforme JADE, par le biais de son outil *snifferAgent* (SnifferJade), offre un moyen aux développeurs pour examiner et visualiser les interactions effectuées entres les agents en temps réels. Ainsi *snifferAgent* utilise une notation graphique qui resemble au diagramme de séquence UML pour traquer les messages échangés. En revanche, la plateforme ZEUS incorpore une trousse à outils nommée : *visualizer* permettant la visualisation, le débogage et l'analyse des sociétés d'agent ZEUS. Le *visionneur* ZEUS a été conçu de manière à mettre l'accent sur les différents aspects de la société à savoir (Collis, Ndumu, Nwana, & Lee, 1998) :

- *Un visionneur de la société* : permet la visualisation de tous les agents et les relations organisationnelles entre eux et également montre l'échange de messages.
- *Un outil de reporting* : montre la décomposition de la société et la distribution des tâches actives et l'état d'exécution des différentes sous-tâches.
- *Un visionneur d'agents* : permet l'observation et le monitoring de l'état interne des agents.
- *Un outil de contrôle* : permet d'examiner et/ou de modifier l'état interne des agents. Ainsi, la redéfinition en temps réel du comportement des agents en modifiant leurs tâches, ressources ou base de données ou encore de modifier la stratégie de coordination. L'outil de contrôle représente un moyen d'administration des agents dans lequel des agents peuvent être créés ou tués ou même les buts associés à un agent peuvent être modifiés.
- *Un outil des statistiques* : montre des statistiques sur les agents et la société dans différents formats.

En ce qui concerne la plateforme COOGAAR, une infrastructure de mesure de performances orienté-agents a été intégrée directement dans l'architecture du système. L'outil de mesure de performance de COOGAAR adopte des métriques dédiées pour mesurer des qualités inhérentes aux SMA. Nous citons à titre d'exemples les métriques correspondant au transport des messages :

- *La langueur de la file d'attente* : désigne le nombre moyen des messages en attente d'envoi.
- *La taille des messages* : désigne le nombre d'octets réservés pour représenter un message.

- *Le nombre de messages* : désigne le nombre de messages envoyés.

Par ailleurs, le monitoring par une tierce partie a été étudié avec parcimonie. Cela est justifié par le nombre très réduit des moniteurs dans cette catégorie. Nous citons à titre d'exemple : AgentSpotter (Doan Van Bien, Lillis, & Collier, 2010a) (Doan Van Bien, Lillis, & Collier, 2010b), AMELI (Esteva, Rosell , Rodriguez-Aguilar , & Arcos, 2004) et également la proposition de (Mani, Garousi, & Far, 2008). AgentSpotter (Doan Van Bien, Lillis, & Collier, 2010b) (Doan Van Bien, Lillis, & Collier, 2010a) offre deux outils de profilage orienté-agent : Agent-Oriented Call-Graph (AOCG) et Space-Time Diagram. AOCG consiste en un moyen de mesure de performance dédié particulièrement pour traquer et analyser les échanges de messages effectués pendant l'exécution du SMA. AOCG illustre graphiquement les échanges des messages en utilisant une notation qui resemble au graphe d'appel de fonction dans la programmation procédurale. Ainsi, AOCG utilise une métrique particulière nommée : *message impact time* permettant d'étudier l'impact d'un message reçu par un agent sur son comportement. Autrement dit, l'impact du message sur le reste des actions effectuées par l'agent récepteur. Cependant, cette métrique reste valable uniquement pour le contexte de AOCG et ne peut plus être généralisée parce qu'il n'existe pas une relation causale entre la réception d'un message et le comportement de l'agent après avoir reçu ce message surtout pour le cas des agents autonomes qui maintiennent dans la majorité des cas des boites aux lettres permettant de sauvegarder les messages reçus et de les gérer par la suite selon leurs propres croyances et buts. En revanche, le deuxième outil nommé *Space-time diagram* consiste en une évaluation de performance du SMA en se basant sur des critères de communication et d'exploitation de ressources (Mémoire et CPU). *Space-time diagram* propose un environnement graphique permettant la visualisation de la métrique adoptée (*message impact time*).

AMELI (Agent-based Middleware for ELectronic Institutions) est une infrastructure qui fait le médiateur (Intergiciel) pour les agents communiquant dans les Institutions Électroniques (IÈ). AMELI est basée sur ISLANDER (Esteva, Cruz, & Sierra, 2002), un éditeur des IÈ permettant la spécification et la vérification des agents médiateurs dans les IÈ. AMELI est supportée par un outil de monitoring permettant l'interception en temps réel de tous les évènements provenant pendant une session d'exécution d'une IÈ à savoir, l'entrée et le départ des agents, les réponses de validation ou de rejets aux différents évènements, les transitions causées par le *timeout*.

La proposition de (Mani, Garousi, & Far, 2008), consiste à mettre l'accent sur le monitoring des SMA dans le but de détecter l'inter-blocage causé par les ressources et la communication. Ainsi, un modèle comportemental a été généré en se basant sur le diagramme de séquence UML.

7. Bilan

Nous allons élaborer dans ce qui suit une étude comparative entre les différentes propositions discutées précédemment dans le but de bien énumérer les lacunes et limites de ces travaux d'une part et, de mettre en valeur notre contribution pour le monitoring des SMA ouverts qui sera présentée dans le chapitre suivant d'autre part. La comparaison proposée a été effectuée selon plusieurs critères à savoir : la technique de profilage utilisée : l'instrumentation JAVA ou AspectJ, la capacité de ces outils de surveiller *les actions performées par les agents*, *le type de monitoring* utilisé : en ligne ou hors lignes vues les inconvénients et les avantages de chaque type, *la prise en considération de l'aspect organisationnel du SMA* particulièrement le modèle AGR, *la prise en considération des performances orientées-agents* (nombre de messages échangés, taille des files d'attente, la surcharge de communication ,etc.), *la modélisation dynamique du comportement du SMA* et finalement nous avons ajouté le critère de *la plateforme agent* dans laquelle le monitoring a été effectué.

En lisant la table suivante (table 3.3), on peut constater que la majorité des solutions de monitoring prennent en considération l'aspect interaction dans les SMA, cela est justifié par le fait que la communication représente le noyau de toutes les activités au sein d'un SMA à savoir : la coordination, la négociation, la collaboration, etc. Ainsi, le monitoring des actions effectuées par les agents a été considéré par tous les travaux à l'exception de (Mani, Garousi, & Far, 2008) et *snifferAgent* de JADE. Mani et al. (Mani, Garousi, & Far, 2008) se focalisent principalement sur l'inter-blocage causé par la communication. Tandis que snifferAgent consiste en la visualisation des messages échangés. En ce qui concerne l'évaluation des performances des SMA, le seul outil qui traite ce point c'est bien *AgentSpotter* conjointement avec *COOGAAR* à la différence que *AgentSpotter* permet un monitoring en tierce partie et *COOGAAR* permet un moyen de monitoring incorporé dans la plateforme agent. Les travaux de monitoring précédents utilisent une technique de profilage basée sur l'instrumentation JAVA tandis que l'instrumentation basée AspectJ n'a pas été encore adoptée pour le contexte des SMA. Finalement, on trouve que le travail proposé dans (Mani, Garousi, & Far, 2008) et (SnifferJade) utilisent le diagramme de séquence UML pour modéliser le comportement dynamique lié à l'échange de messages par contre le travail proposé dans (Nwana, Ndumu, Lee, & Collis, 1999) utilise la technique de visualisation basée sur les enregistrements vidéos.

Cependant, on constate que l'aspect organisationnel n'a pas été exploré clairement dans les propositions précédentes. À l'exception de (Esteva, Rosell , Rodriguez-Aguilar , & Arcos, 2004) qui utilise un outil de monitoring dédié aux IÈ dans lesquelles les SMA sont implémentés en société d'agents partitionnés en groupes. De même, dans (Esteva, Rosell , Rodriguez-Aguilar , & Arcos, 2004) le mouvement des agents exprimé en termes d'entrée et de départ, et l'achèvement des buts pour des agents ont été surveillés.

CRITÈRES / OUTIL	PLATEFORME AGENT	COLLECTE DES INFORMATIONS SPÉCIFIQUES AUX SMA	ECHANGE DE MESSAGES	ÉVALUATION DES PERFORMANCES DU SMA	ÉVÉNEMENTS BASÉS -AGR	MONITORING EN TEMPS REEL	ACTIONS EFFECTUÉES	TECHNIQUE DE PROFILAGE	MODÈLE COMPORTEMENT-ALE
AgentSpotter	**Agent Factory**	√	√	√			√	**Instrumentati -on JAVA**	**Graphe d'appel orienté agent**
Esteva et al.	**JADE**		√			√	√		
Mani et al.			√			√			**Diagramme de séquence UML**
Sniffer JADE	**JADE**		√			√		**Instrumentat-ion JAVA**	**Diagramme de séquence UML**
Zeus toolKit	**ZEUS**		√			√	√	**Instrumentat-ion JAVA**	**Outil vidéo**

Table 3.3 : Résumé des approches de monitoring des SMA. (Chebout, Mokhati, Badri , & Babahenini, 2019).

8. Conclusion

Le présent chapitre a été consacré à la présentation des concepts et notions liés au monitoring des SMA. Nous avons introduit des définitions d'ordre général des concepts de base concernant le monitoring, le profilage et l'instrumentation. Une comparaison entre les deux termes de monitoring et profilage a été élaborée dans le but de bien positionner notre vision pour le choix de la technique appropriée pour notre contexte d'étude. Ainsi, nous avons adopté le terme monitoring au lieu de profilage pour des raisons bien justifiées.

Nous avons également mis en évidence le langage AspectJ comme étant une nouvelle technique utilisée pour l'instrumentation des systèmes. AspectJ sera maintenu dans notre étude des SMA grâce à sa flexibilité d'intégration avec le langage JAVA et ses performances remarquables en termes de coût de surcharge réduit. À l'instar du chapitre précédent, une étude comparative a été montrée et discutée selon plusieurs critères techniques. Cette étude nous aide énormément à surpasser les lacunes observées pour notre proposition de monitoring qui sera le sujet du chapitre suivant. Un outil de monitoring basé sur la POA pour l'analyse et l'étude des SMA ouverts basés-AGR sera bien exposé dans le prochain chapitre.

Chapitre 4

UNE APPROCHE DE MONITORING DES SYSTÈMES MULTI-AGENTS OUVERTS

Sommaire

1. Introduction

Dans ce chapitre, nous présentons notre première contribution sur le monitoring des SMA ouverts basés-AGR. L'approche de monitoring consiste en l'observation, l'examen et l'analyse dynamique du comportement d'un SMA ouvert pendant son exécution. Ainsi, notre proposition utilise les techniques de la POA pour le profilage et l'instrumentation. Par conséquent, maintes informations pertinentes ont été récoltées à savoir, informations sur le cycle de vie des agents, informations sur la communication des agents, informations sur les actions effectuées par les agents et également des informations sur les groupes créés, détruis et les rôles accordés et rejetés pour chaque groupe. Une modélisation dynamique des interactions effectuées en temps réel a été générée en conséquence. L'approche proposée est supportée par un outil logiciel nommé : RT-MTOMAS pour : Real-time Monitoring Tool for Open Multi-Agent Systems, et est validée par une étude de cas sous la plateforme MaDKit.

Nous commençons ce chapitre par la présentation de notre proposition et l'architecture de RT-MTOMAS, ensuite nous discutons les différents aspects techniques de l'outil développé. Finalement, nous montrons à travers une étude de cas l'applicabilité de notre approche et son apport par rapport aux travaux discutés dans le chapitre précédent.

2. Approche proposée

Une approche initiale de contrôlabilité (Chebout, Mokhati, Badri, & Babahenini, 2016) consiste à diviser le processus de contrôle en deux parties essentielles (figure 4.1), à savoir : le monitoring et le contrôle proprement dit. Les deux parties représentent des étapes causales et complémentaires. Par ailleurs, le monitoring précède nécessairement le contrôle et également la phase de contrôle n'aura pas lieu si la phase de monitoring ne finit pas par fournir un rapport (un profil) comportant une analyse globale du contexte d'exécution du SSM.

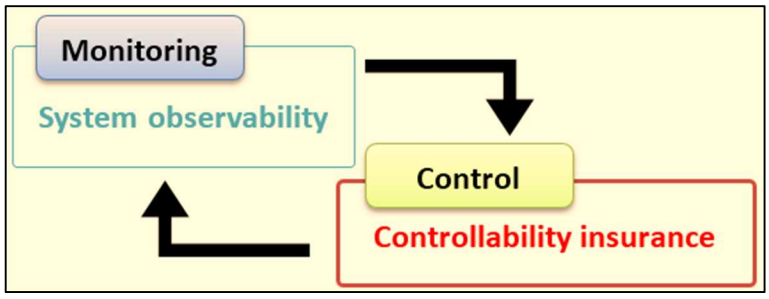


Figure 4.1 : Le processus de contrôle.

L'objectif principal de l'approche de monitoring proposée (Chebout, Mokhati, Badri , & Babahenini, 2019) est d'étendre le SSM par un ensemble d'aspects dans le but de récupérer les informations dont on a besoin (figure 4.2) dans la deuxième étape du processus de contrôle.

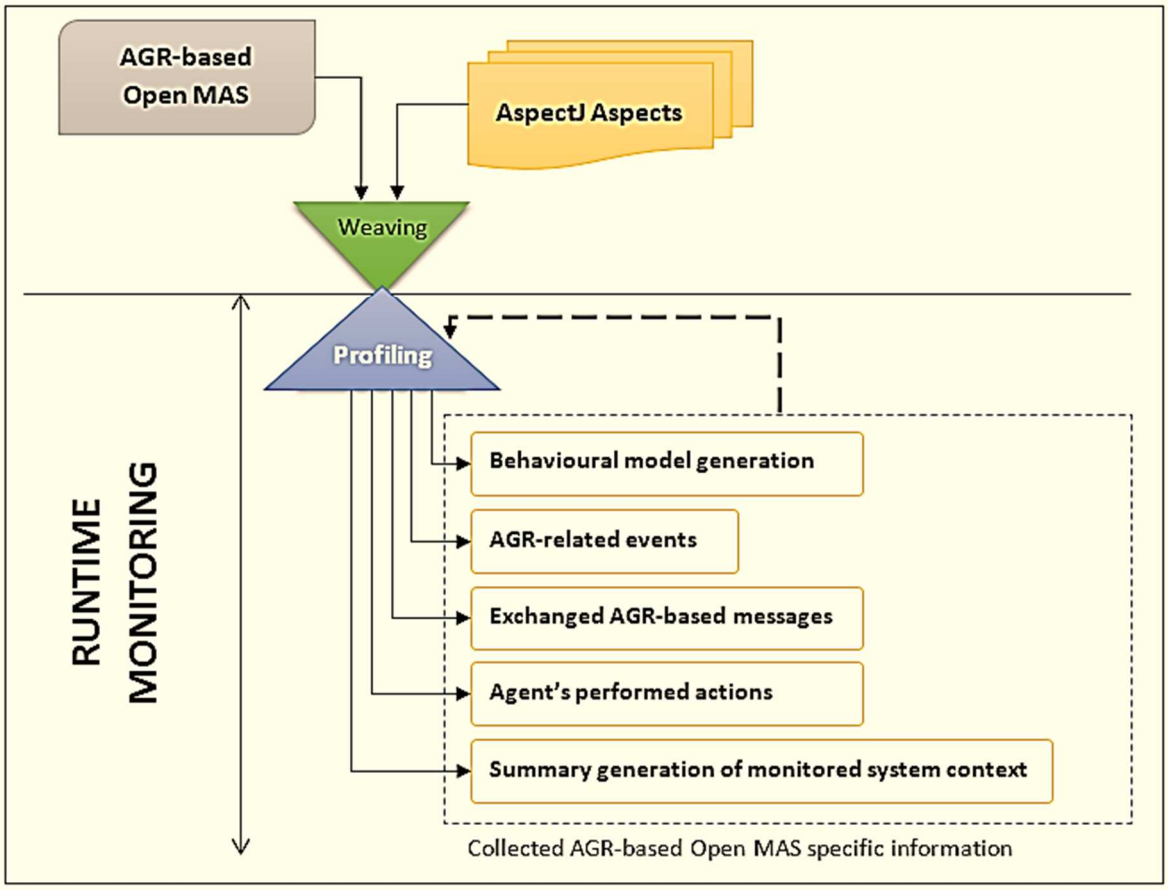


Figure 4.2 : La méthodologie de notre approche proposée(Chebout, Mokhati, Badri , & Babahenini, 2019).

Les aspects s'intègrent avec le code de l'application en cours de monitoring en utilisant le mécanisme de tissage qui consiste à intercaler le code original avec les coupes définies dans l'aspect. Ensuite, on va collecter les informations désirées par le biais du profilage (figure 4.2). Ce dernier consiste à récupérer des informations pertinentes à savoir :

- La liste des actions effectuées par un agent donné,
- Le suivi des cycles de vie de chaque agent en termes des sections exécutées par cet agent. Selon la documentation officielle de MaDKit (Gutknecht & Ferber, 2000), le cycle de vie d'un agent passe par trois sections différentes :
  - *Activate* : une fois l'agent entre dans le système
  - *Live* : c'est la méthode principale qui décrit le comportement à exécuter par l'agent,
  - *End* : dans cette section, l'agent va exécuter une suite d'instructions puis il quitte le système.
- L'interception des évènements en relation avec le modèle AGR à savoir, la création et la destruction des groupes, la liste des rôles demandés et rejetés durant l'exécution du système, la liste des rôles dans chaque groupe, etc.
- La liste des messages échangés.

- Une modélisation comportementale sera générée en temps réel décrivant les interactions effectuées par les agents. Le modèle généré est exprimé en termes de diagrammes de séquence d'agents AUML.

Par ailleurs, RT-MTOMAS est défini comme étant un intergiciel (en anglais : *middleware*) qui se positionne entre la plateforme agent et le SSM (figure 4.3). Ainsi, l'architecture de RT-MTOMAS est conçue d'une manière à créer un réseau d'échange d'informations entre MaDKit et le SMA.

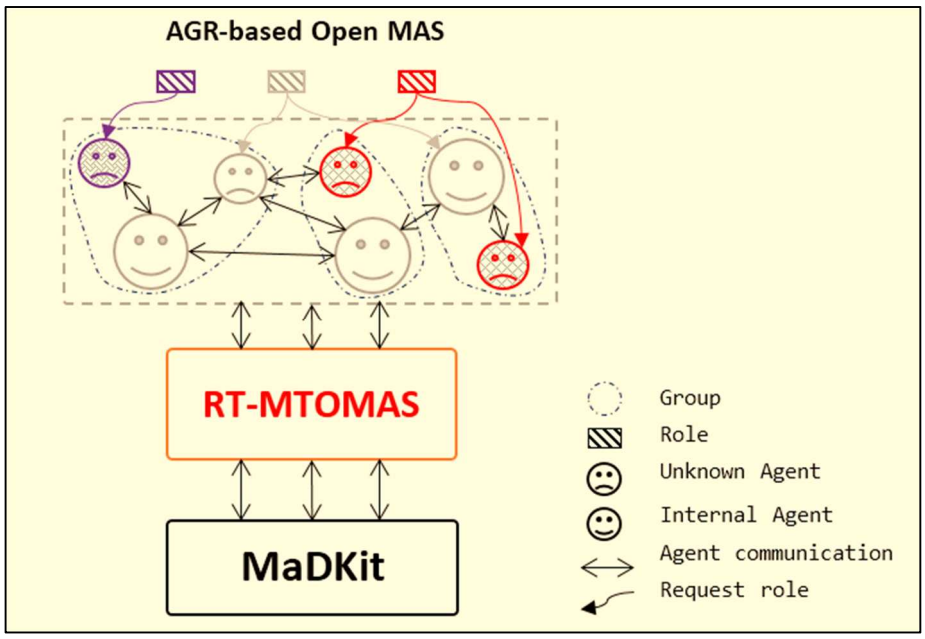


Figure 4.3 : La médiation de RT-MTOMAS via MaDKit(Chebout, Mokhati, Badri , & Babahenini, 2019).

3. Architecture de RT-MTOMAS

L'idée fondamentale avec laquelle RT-MTOMAS est conçu est la proposition d'une solution logicielle qui va s'exécuter parallèlement entre une plateforme agent basée-AGR et le SSM afin d'observer, d'une manière permanente, son comportement. Tel que mentionné dans le chapitre précédent, le moyen le plus répandu dans la littérature pour l'instrumentation du code source est l'utilisation des techniques de la POA. Ainsi, la POA et particulièrement le langage AspectJ a été utilisé récemment à des fins de profilage dans maints travaux de recherches (Nusayr & Cook, 2009) (Avgustinov, et al., 2006) (Chen & Roşu, 2005) (Bodden & Havelund , 2008) (Richters & Gogolla, 2003). En revanche, un nombre très restreint de travaux ont mis l'accent sur le monitoring orienté-agent où les spécifiés des SMA ont été prises en considération.

À cet effet, l'architecture de RT-MTOMAS est composée d'un moteur d'aspect (figure 4.4) qui est responsable de récupérer les informations pertinentes.

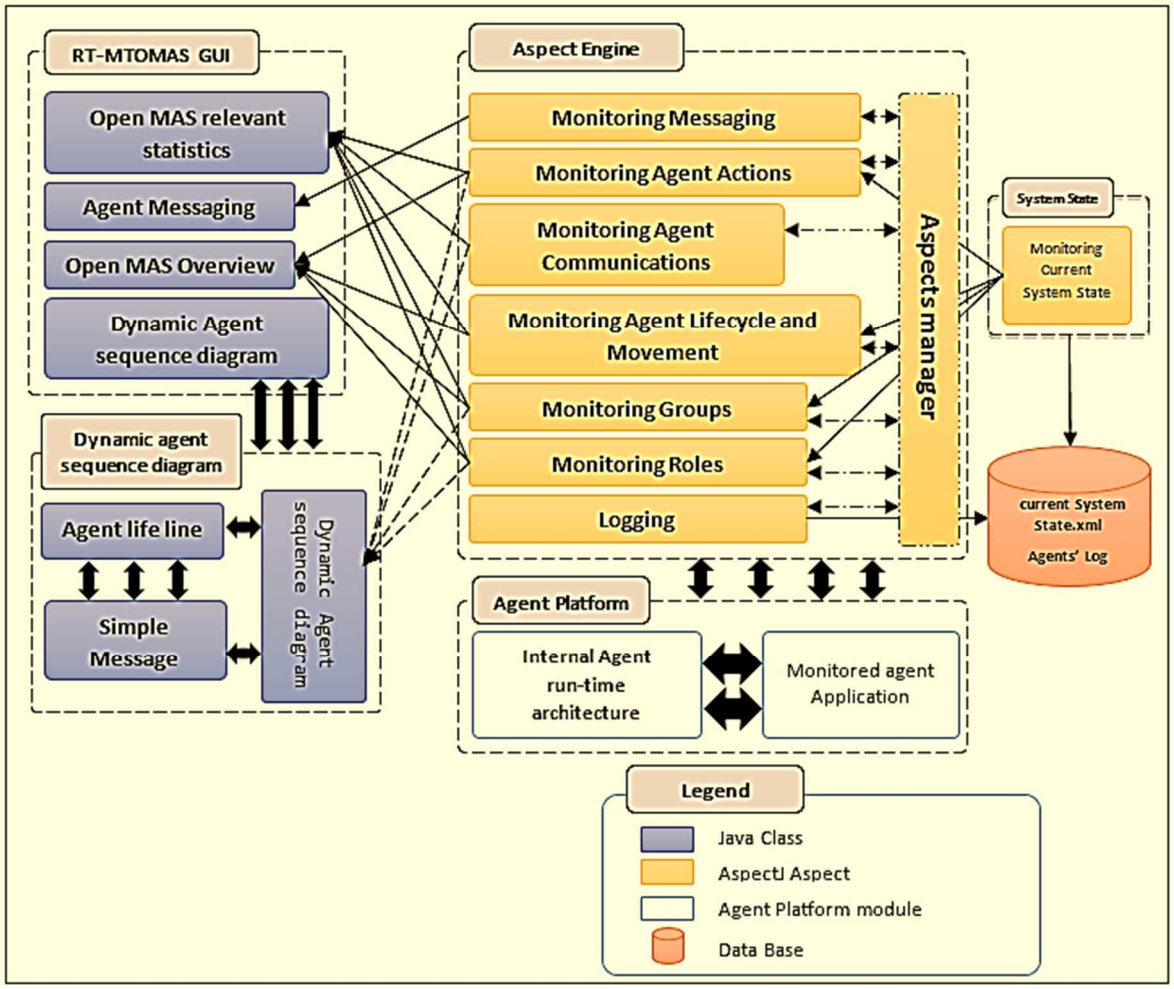


Figure 4.4 : L'architecture de RT-MTOMAS (Chebout, Mokhati, Badri , & Babahenini, 2019).

Ainsi, le moteur d'aspects formule un ensemble de requêtes empaquetées dans des modules (aspects) pour intercepter les évènements provenant de la plateforme agent à savoir :

- **Le monitoring de la messagerie** (Monitoring Messaging) : ce module consiste à récupérer des informations liées à l'échange de messages en interceptant les différentes primitives dédiées à l'échange de messages. Pour cela, un résumé de communication sera élaboré comportant :
  - ConvID : l'identifiant de la conversation,
  - Sender : l'identifiant de l'agent émetteur,
  - Receiver : l'identifiant de l'agent récepteur,
  - MsgContent : le contenu du message échangé,
  - MethodeSignature : l'entête de la primitive utilisée pour l'échange du message.
  - LineCodeNumber : le numéro de la ligne de code source qui correspond à la primitive utilisée pour envoyer le message.

Dans le cas de la plateforme MaDKit, la table suivante (table 4.1) résume les primitives qu'un développeur peut utiliser pour exprimer l'échange de messages :

Prefix	Method	N°	Signature
sendMessage	**sendMessage**	1	**sendMessage**(AgentAddress receiver, Message messageToSend)
		2	**sendMessage**(String community, String group, String role, Message message)
	sendMessageWithRole	3	**sendMessageWithRole**(String community, String group, String role, Message message, String senderRole)
	sendMessageWithRole AndWaitForReply	4	**sendMessageWithRoleAndWaitForReply**(AgentAddress receiver, Message messageToSend, String senderRole)
		5	**sendMessageWithRoleAndWaitForReply**(AgentAddress receiver, Message messageToSend, String senderRole, int timeOutMilliSeconds)
		6	**sendMessageWithRoleAndWaitForReply**(String community, String group, String role, Message messageToSend, String senderRole)
		7	**sendMessageWithRoleAndWaitForReply**(String community, String group, String role, Message messageToSend, String senderRole, int timeOutMilliSeconds)
	sendMessageAndWait ForReply	8	**sendMessageAndWaitForReply**(AgentAddress receiver, Message messageToSend)
		9	**sendMessageAndWaitForReply**(AgentAddress receiver, Message messageToSend, int timeOutMilliSeconds)
		10	**sendMessageAndWaitForReply**(String community, String group, String role, Message messageToSend)
		11	**sendMessageAndWaitForReply**(String community, String group, String role, Message messageToSend, int timeOutMilliSeconds)

Table 4.1 : Résumé sur les primitives de messagerie basées AGR de MaDKit.

La particularité de la plateforme MaDKit comme étant une plateforme agent basée-AGR, réside dans la manière avec laquelle l'agent s'interagit avec autrui. Ainsi, un agent sous MaDKit doit mentionner son groupe et son rôle dans le message à échanger, dans six primitives (table 4.1, lignes : 2, 3, 6, 7, 10, 11), et son rôle uniquement dans deux primitives (table4.1, lignes : 4 et 5), on parle dans ce cas sur l'interaction basée-AGR. Cependant, aucune information sur le groupe ou le rôle de l'agent ne doit pas être communiquée dans les autres primitives (table 4.1, lignes : 1, 8 et 9) qui est le cas de la majorité des plateformes existantes.

Par ailleurs, l'interaction basée-AGR a été justifiée par le fait que les agents peuvent se communiquer seulement s'ils appartiennent dans le même groupe.

Dans le contexte de RT-MTOMAS, l'interaction basée-AGR figure clairement dans le modèle comportemental généré (voire section n°4).

- **Le monitoring des actions d'agents** (Monitoring Agent Actions) : dans ce module toutes les actions effectuées par un agent seront interceptées et enregistrées dans une structure de données particulière. Ainsi, les actions possibles qu'un agent peut effectuer pour le cas de MaDKit ont été résumées dans la table suivante (table 4.2) :

AGR	Action	
Agent	1	launchAgent
	2	executeThisAgent
	3	Reload
	4	getMyRoles
	5	getExistingRoles
	6	checkAgentAddress
	7	destroyCommunity
	8	launchXmlAgents
	9	killAgent
	10	getAgentWithRole
	11	getAgentsWithRole
	12	getDistantAgentWithRole
	13	getExistingCommunities
	14	getExistingGroups
	15	getMyGroups
Groupe	16	createGroup
	17	createGroupIfAbsent
	18	leaveGroup
	19	destroyGroup
	20	bucketModeCreateGroup
Rôle	21	requestRole
	22	leaveRole
	23	bucketModeRequestRole
	24	destroyRole

Table 4.2 : Primitives AGR de MaDKit.

Entre autres, un agent peut créer ou détruire des groupes (table 4.2, lignes : 16 et 17), demander des rôles (table 4.2, ligne : 21) et même lancer d'autres agents (table 4.2, lignes : 1 et 2) ou encore récupérer des informations sur le contexte d'exécution comme la liste des groupes dont il est membre (tables 4.2, lignes : 14 et 15) ou les rôles qui lui sont affectés (tables 4.2, lignes : 10, 11 et 12). Par conséquent, les actions effectuées par un agent contribueront dans la génération d'un profil d'exécution pour chaque agent et pour le système entier.

- **Le monitoring de la communication des agents** (Monitoring Agent communications) : ce module est responsable de la génération d'un modèle comportemental dynamique. Dans le cadre de cette thèse, nous avons adopté le diagramme de séquence d'agent AUML. Ainsi, le diagramme généré est nommé RT-MTOMAS DASD pour : RT-MTOMAS Dynamic Agent Sequence Diagram. Nous allons utiliser dans le reste de ce chapitre l'acronyme RT-MTOMAS DASD au lieu de la dénomination étendue pour des raisons d'expressivités et de concision.

- **Le monitoring de cycle de vie des agents et leurs mouvements** (Monitoring agent lifecycle and movements) : dans ce module le mouvement des agents en termes d'arrivée et de départ ainsi que les différentes sections exécutées (*activate*, *live* et *end*) seront interceptés. Pour plus de clarification, la figure suivante (figure 4.5) montre une portion de code d'un aspect qui intercepte la section *live* d'un agent sous MaDKit :

```
1  pointcut livesAgent() : execution(* *..live(..));
2
3      before() : livesAgent(){
4          AbstractAgent agAd = (AbstractAgent) thisJoinPoint.getTarget();
5          if (!createdAgentList.contains(agAd)){
6              createdAgentList.add(agAd);
7          }
8      }
```

Figure 4.5 : L'interception de l'exécution de la section *live*.

La coupe *livesAgent* (figure 4.5, ligne 1), intercepte tous les appels à la primitive *live*, ensuite une liste nommée *createdAgentList* comportant tous les agents exécutant la section *live* est maintenue.

- **Le monitoring des groupes** (Monitoring groups) : ce module permet de traquer dynamiquement la création et la destruction des groupes et d'établir, en conséquence, un résumé comportant des informations en relation. À cet effet, la table suivante (table 4.3) liste les réponses possibles fournies par MaDKit à une requête de type *createGroup* ou *requestRole* :

Code retourné	Raison de l'exception
NOT_COMMUNITY	Le groupe n'existe pas.
NOT_GROUP	Le rôle n'existe pas.
NOT_ROLE	L'agent n'est pas dans ce groupe.
NOT_IN_GROUP	L'agent possède déjà le rôle demandé.
ROLE_ALREADY_HANDLED	Échec de la demande d'un rôle dans un groupe sécurisé.
ACCESS_DENIED	L'agent ne possède pas le rôle qu'il est censé avoir faire une action particulière, e.g. sendMessageWith Role.

ROLE_NOT_HANDLED	Le group existe déja.
NO_RECIPIENT_FOUND	La primitive requestRole (ou createGroup) est utilisée dans la section activate et que l'agent a été lancé en utilisant la primitive launchAgentBucket.

Table 4.3 : Exceptions AGR générées par MaDKit.

Ainsi, un agent ne doit pas demander d'effectuer un rôle dans un groupe inexistant, ou de créer un groupe qui est déjà créé.

- **Le monitoring des rôles** (Monitoring rôles) : ce module a pour objectif l'observation de l'acquisition ou le rejet des rôles pour un agent. Dans le contexte de la plateforme MaDKit (à l'instar de la plateforme Magentix2), un agent qui veut entrer dans le système doit exprimer sa demande explicitement en utilisant la primitive *requestRole* (figure 4.6) :

```
public int requestRole(String communityName,
                       String groupName,
                       String roleName)
```

Figure 4.6 : La signature de la primitive requestRole.

Ainsi, la méthode *requestRole* retourne les valeurs suivantes (table 4.4) :

Valeur retournée	Signification
1	Opération réussie.
-1	Accès refusé.
2	Le rôle est déjà manipulé par cet agent.
3	Le groupe n'existe pas.
4	La communauté n'existe pas.

Table 4.4 : Les valeurs retournées par la méthode requestRole(Gutknecht & Ferber, 2000).

La figure suivante (figure 4.7) montre une portion de code d'un aspect interceptant la méthode *requestRole* :

```
1  pointcut observeRequiredRoleV2(String communityName, String groupName, String Name, Object passKey):
2      call(ReturnCode *.requestRole(..)) && args(communityName, groupName, roleName, passKey);

3      after (String communityName, String groupName, String roleName, Object passKey) returning (
       ReturnCode r): observeRequiredRoleV2(communityName, groupName, roleName, passKey) {
4
5          AbstractAgent agAd = (AbstractAgent) thisJoinPoint.getTarget();
6          if (!listOfAgentThatPlayinARole.contains(agAd)){
7              listOfAgentThatPlayinARole.add(agAd);
8              agentTiming.put(agAd.
9                  getAgentAddressIn(communityName, groupName, roleName).getAgentNetworkID(), MonitorGui
                   .appTime.getText());}
10
11         List<Integer> values = new ArrayList<Integer>();
12         Map<String, ReturnCode> t = new HashMap<String, ReturnCode>();
13     if (r.equals(ReturnCode.SUCCESS)){
14         if (!ListOfRole.containsKey(groupName)){
15             t.putIfAbsent(roleName , r);
16             ListOfRole.put(groupName, t);
17         }
18             //...
```

Figure 4.7 : L'interception de la primitive requestRole(Chebout, Mokhati, Badri , & Babahenini, 2019).

De même des structures de données ont été maintenues pour la gestion des rôles acceptés ou rejetés avec toutes les informations pertinentes.

- **Le logging** (logging) : ce module est responsable de l'élaboration d'un aperçu général sur le contexte de l'exécution comportant tous les évènements de type LOG provenant des agents pendant la session de profilage.

- **Le gestionnaire des aspects** (Aspect manager) : cet aspect consiste en la gestion de la précédence entre les aspects, dans le but d'imposer une priorité dans l'exécution des aspects selon le déroulement de l'exécution.

4. Interface RT-MTOMAS

L'interface de l'outil RT-MTOMAS est montrée dans la figure suivante (figure 4.8). Les captures écrans 1, 2, 3 et 4 correspondent à : une vue générale du SSM, statistiques AGR, la messagerie des agents et le diagramme de séquence d'agent dynamique respectivement. Nous allons détailler dans ce qui suit la fonctionnalité de chaque module :

- Capture d'écran n°1 (*la vue générale du SSM*) : cette capture d'écran présente en temps réel un résumé des évènements basés-AGR tels que : les rôles demandés, les groupes dans lesquels les rôles sont joués, le nom de la communauté dans laquelle les groupes sont créés et également la réponse de MaDKit pour chaque type d'évènement (table 4.3). Plus de détails sur les exceptions générées par MaDKit seront discutés dans la section étude de cas. D'autres informations statistiques telles que : le nombre de groupes créés (ou détruis), le nombre des demandes de rôles acceptés (ou rejetés), etc. sont fournies.

- Capture d'écran n°2 (*les statistiques AGR*) : dans cette capture d'écran des diagrammes en bâtons et des diagrammes circulaires sont créés automatiquement pour illustrer graphiquement les groupes créés et les rôles joués dans chaque groupe et également le taux d'acceptation ou de rejet des rôles dans chaque groupe.

- *Capture d'écran n°3* (*la messagerie des agents*) : des informations sur la communication des agents sont affichées dans cette capture d'écran à savoir, l'agent appelant, l'agent appelé, l'identifiant de la conversation, le contenu du message échangé, la primitive MaDKit utilisée dans cette conversation et le numéro de la ligne de code qui correspond à cette primitive.

- *Capture d'écran n°4* (*RT-MTOMAS DASD*) : cette capture d'écran montre le diagramme RT-MTOMAS DASD généré et édité automatiquement au cours de l'exécution du SSM. La visualisation est complètement interactive, le développeur peut faire des manipulations de zoom, de déplacement à l'intérieur du diagramme et également des défilements. Ainsi, la particularité de RT-MTOMAS DASD vient du fait que des concepts AGR ont été pris en considération lors de la génération. À titre d'exemple, la ligne de vie de l'agent (agent life line) est étiquetée par l'identificateur de l'agent en concaténation avec son rôle, par exemple : (4@863:broker) et (2@863:broker) sont deux agents jouant le rôle d'un courtier. Ainsi, les lignes de vie des agents sont colorées de manière à différencier les groupes d'agents. Les lignes de vie colorées avec la même couleur correspondent aux agents jouant des rôles dans le même groupe. Par ailleurs, les messages dans RT-MTOMAS sont étiquetés par deux informations à savoir, l'identificateur de la conversation et le contenu du message. L'identificateur de la conversation offre un moyen de refactoring aux développeurs en cherchant cet identificateur dans l'historique de conversation (capture d'écran n°3) puis on va récupérer le numéro de la ligne de la primitive dans le code source. Cela accélère le moyen de profilage si des éventuelles modifications dans le code source du SSM auront lieu.

Il est également possible de récupérer l'ensemble de la configuration du SSM à tout moment durant l'exécution. Cette fonctionnalité de RT-MTOMAS est nommée *getCurrentSystemState* et déployée sous format XML dans le but de procéder à une comparaison entre l'état actuel du système avec la cible.

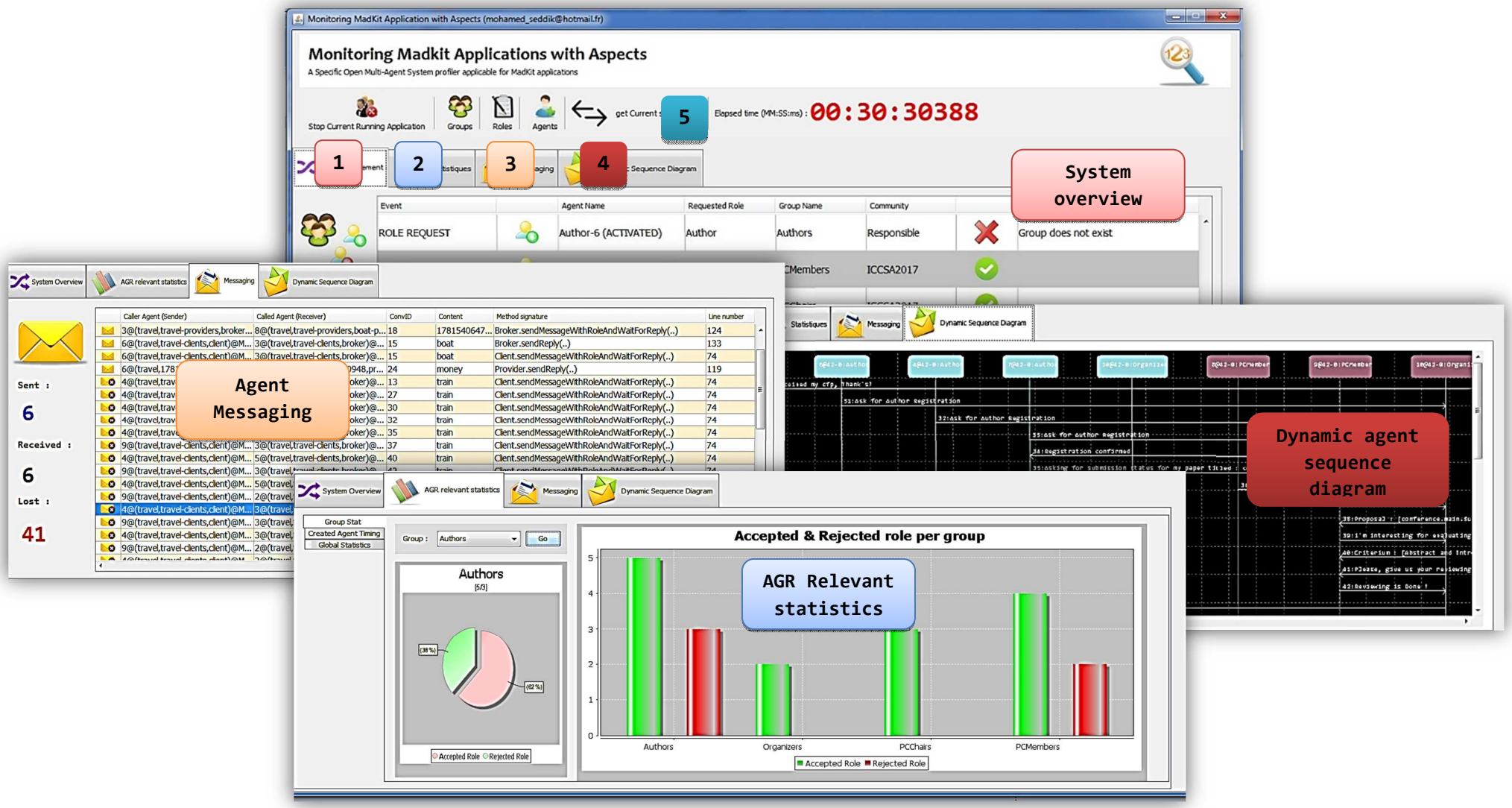


Figure 4.8 : Des captures d'écran de RT-MTOMAS (Chebout, Mokhati, Badri , & Babahenini, 2019).

5. Étude de cas

Dans le but de démontrer l'applicabilité de RT-MTOMAS comme étant un moyen de monitoring des SMA ouverts basé-AGR, nous avons traité une étude de cas déployée avec la plateforme MaDKit sous le nom : agence de voyages (*travel agency*) qui représente, d'après ses créateurs (Ferber, Gutknecht, & Michel, 2004), une simplification d'une application de commerce électronique (e-commerce).

Ce système comporte trois types d'agents : client, courtier, et fournisseur. Le client entre, via une demande explicite, dans le groupe des clients dans le but d'acheter un ticket de voyage de l'un des moyens de transport suivant : train, bateau ou bus. Pour cela, le client interroge le courtier pour en avoir un ticket. Ensuite, le courtier, à son tour, interroge le fournisseur via un appel à proposition (CFP : Call For Proposal). Le fournisseur va chercher le moyen de transport disponible pour vendre ses tickets. Si c'est le cas, un groupe nommé *contrat* sera créé par le fournisseur comportant le client et le fournisseur jouant les nouveaux rôles : acheteur et vendeur respectivement. Si le contrat a été finalisé, le client acheteur lance un autre client et quitte le système, sinon il cherche un autre moyen de transport. La figure suivante (figure 4.9) montre le flot de contrôle de client exprimé en termes de diagramme d'activité UML :

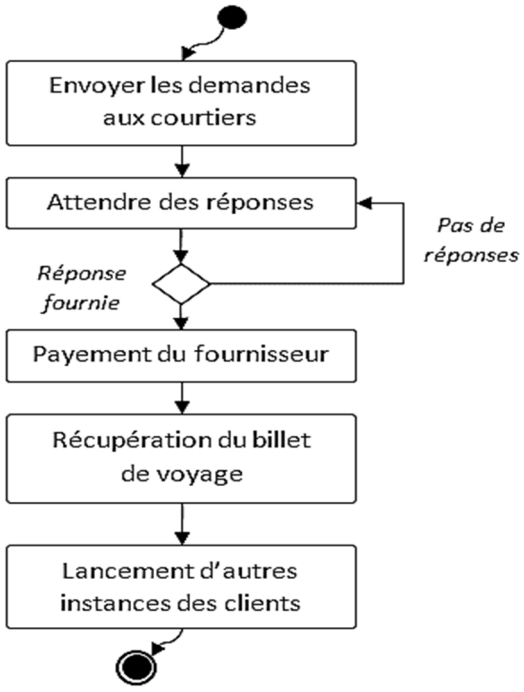


Figure 4.9 : Le diagramme d'activité du client.

L'agent courtier se place comme étant l'intermédiaire entre le client et le fournisseur et joue à la fois le rôle d'un courtier dans le groupe des clients, des fournisseurs et éventuellement dans le groupe des courtiers. Sa tâche principale est d'interroger le

fournisseur pour avoir un ticket d'un moyen de transport et de sélectionner la meilleure offre proposée. La figure suivante (figure 4.10) montre le diagramme d'activité UML du courtier :

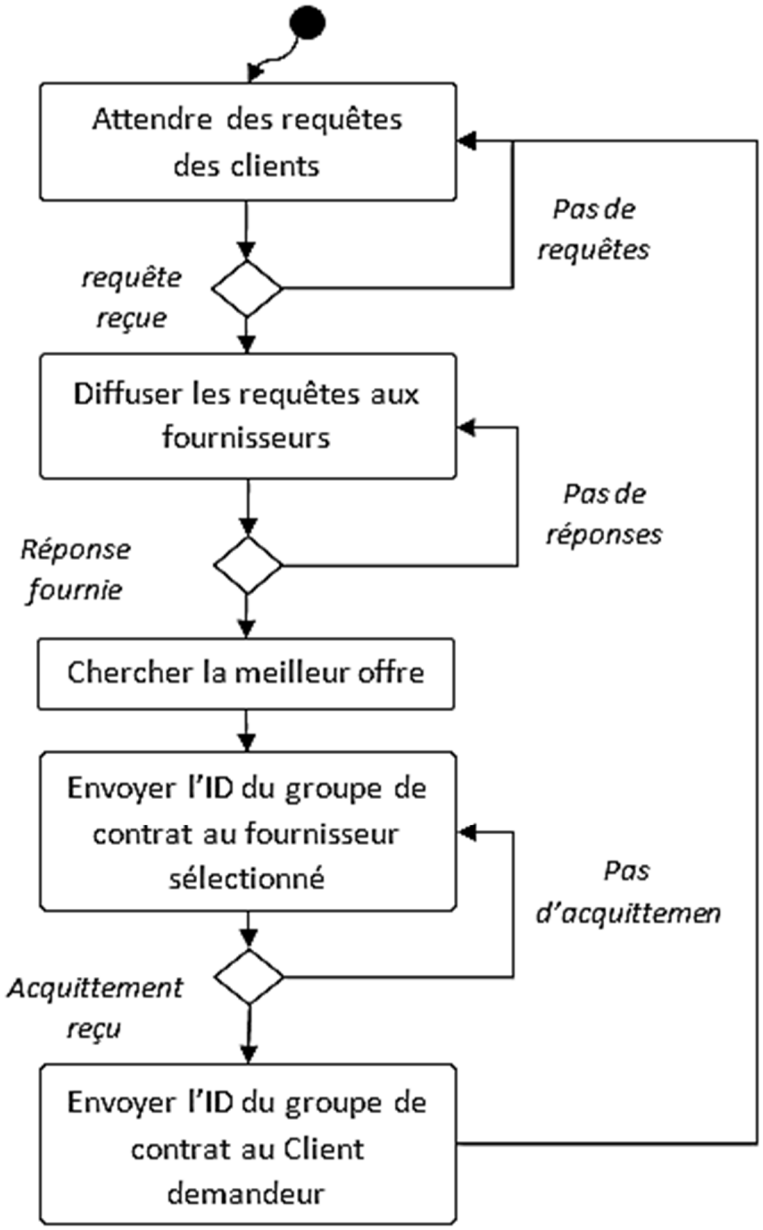


Figure 4.10 : Le diagramme d'activité du courtier.

En ce qui concerne l'agent fournisseur (figure 4.11), sa mission se résume à la recherche d'un moyen de transport disponible pour vendre ses tickets. À l'arrivée d'une demande de la part des courtiers concernant une requête d'un moyen de transport par un client, il propose le prix du ticket du moyen de transport trouvé. Sinon, si la demande concerne une confirmation d'achat de ticket, le fournisseur va créer un groupe contrat en jouant le rôle d'un vendeur.

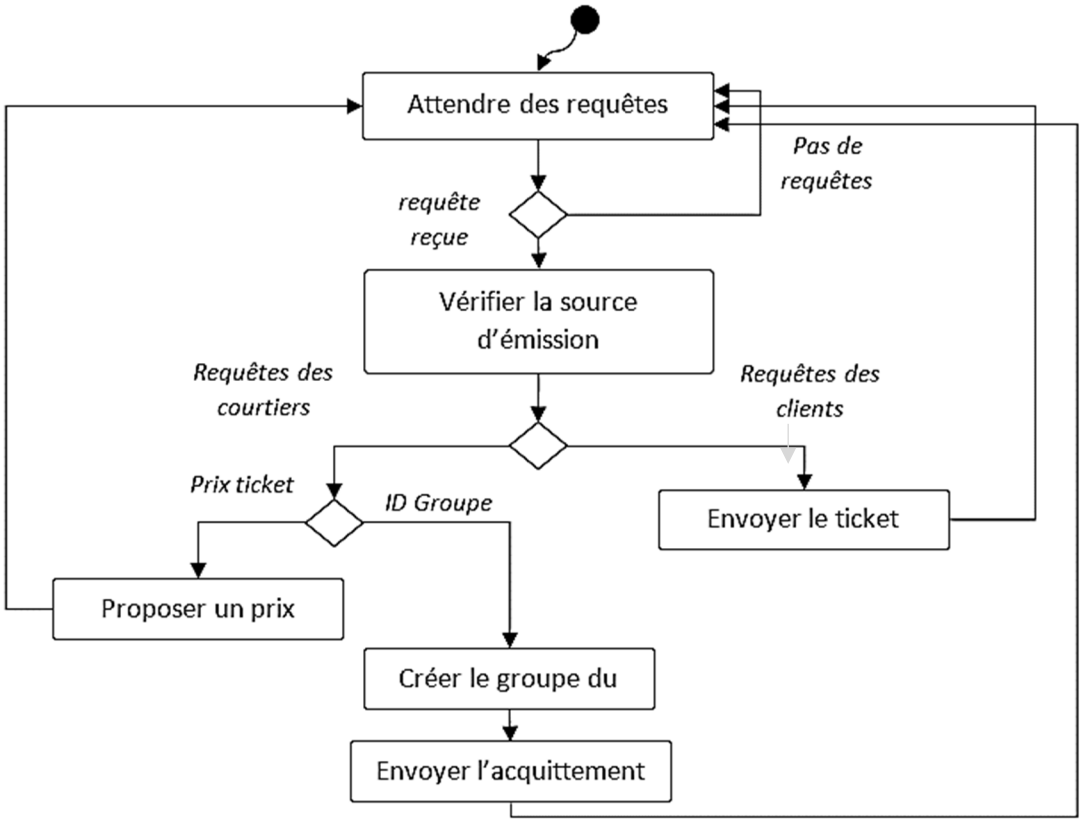
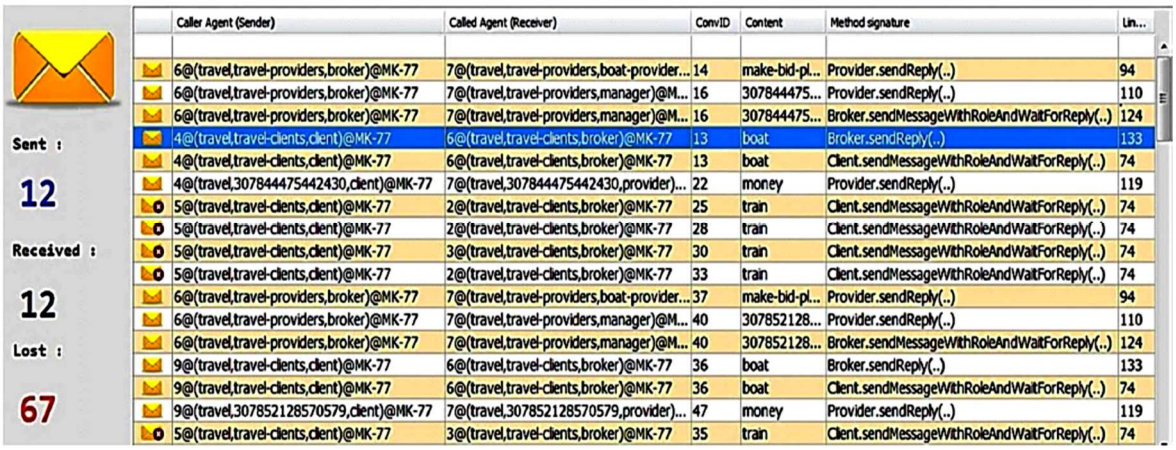


Figure 4.11 : Le diagramme d'activité du fournisseur.

Les trois groupes : clients, fournisseurs et courtiers sont créés par les premiers agents entrant dans le système à savoir : le client, le fournisseur et le courtier respectivement. Les autres agents qui viennent par la suite vérifient dans un premier lieu l'existence des groupes qui leur conviennent. Si c'est le cas, ils passent des demandes explicites pour jouer des rôles dans ces groupes sinon ils procèdent à la création de ces groupes. Sous MaDKit, un groupe est créé pour la première fois par le biais de la primitive *createGroupe*. Pour vérifier si le groupe a été déjà créé on utilise la primitive *createGroupIfAbsent*. De manière similaire, un agent qui veut jouer un rôle dans un groupe donné doit s'exprimer en utilisant la primitive *requestRole* en mentionnant le nom du groupe.

Initialement, nous avons lancé le système avec le paramétrage suivant : trois clients, deux courtiers et sept fournisseurs. Une fois le client possède son ticket il lancera 4 autres instances du client avant de quitter le groupe des clients en utilisant la primitive *launchAgent* (table 4.2). La figure suivante (figure 4.12) montre que le client (@5) a envoyé quatre messages en utilisant la primitive *sendMessageWith RoleAndWaitForReply* (conversationID n° 25, 28, 30 et 33) en demandant à l'agent courtier un ticket pour un train et comme l'agent courtier a été déjà préoccupé à la fois par le traitement des demandes des autres clients et l'interrogation des fournisseurs, les messages des clients n'ont pas été traités et sont considérés comme perdus (timeout consommé).



Caller Agent (Sender)	Called Agent (Receiver)	ConvID	Content	Method signature	Lin...
6@(travel,travel-providers,broker)@MK-77	7@(travel,travel-providers,boat-provider...	14	make-bid-pl...	Provider.sendReply(..)	94
6@(travel,travel-providers,broker)@MK-77	7@(travel,travel-providers,manager)@M...	16	307844475...	Provider.sendReply(..)	110
6@(travel,travel-providers,broker)@MK-77	7@(travel,travel-providers,manager)@M...	16	307844475...	Broker.sendMessageWithRoleAndWaitForReply(..)	124
4@(travel,travel-clients,client)@MK-77	6@(travel,travel-clients,broker)@MK-77	13	boat	Broker.sendReply(..)	133
4@(travel,travel-clients,client)@MK-77	6@(travel,travel-clients,broker)@MK-77	13	boat	Client.sendMessageWithRoleAndWaitForReply(..)	74
4@(travel,307844475442430,client)@MK-77	7@(travel,307844475442430,provider)...	22	money	Provider.sendReply(..)	119
5@(travel,travel-clients,client)@MK-77	2@(travel,travel-clients,broker)@MK-77	25	train	Client.sendMessageWithRoleAndWaitForReply(..)	74
5@(travel,travel-clients,client)@MK-77	2@(travel,travel-clients,broker)@MK-77	28	train	Client.sendMessageWithRoleAndWaitForReply(..)	74
5@(travel,travel-clients,client)@MK-77	3@(travel,travel-clients,broker)@MK-77	30	train	Client.sendMessageWithRoleAndWaitForReply(..)	74
5@(travel,travel-clients,client)@MK-77	2@(travel,travel-clients,broker)@MK-77	33	train	Client.sendMessageWithRoleAndWaitForReply(..)	74
6@(travel,travel-providers,broker)@MK-77	7@(travel,travel-providers,boat-provider...	37	make-bid-pl...	Provider.sendReply(..)	94
6@(travel,travel-providers,broker)@MK-77	7@(travel,travel-providers,manager)@M...	40	307852128...	Provider.sendReply(..)	110
6@(travel,travel-providers,broker)@MK-77	7@(travel,travel-providers,manager)@M...	40	307852128...	Broker.sendMessageWithRoleAndWaitForReply(..)	124
9@(travel,travel-clients,client)@MK-77	6@(travel,travel-clients,broker)@MK-77	36	boat	Broker.sendReply(..)	133
9@(travel,travel-clients,client)@MK-77	6@(travel,travel-clients,broker)@MK-77	36	boat	Client.sendMessageWithRoleAndWaitForReply(..)	74
9@(travel,307852128570579,client)@MK-77	7@(travel,307852128570579,provider)...	47	money	Provider.sendReply(..)	119
5@(travel,travel-clients,client)@MK-77	3@(travel,travel-clients,broker)@MK-77	35	train	Client.sendMessageWithRoleAndWaitForReply(..)	74

Figure 4.12 : Les messages envoyés, reçus et perdus.

Les messages rejetés montrent clairement que les courtiers (ou les fournisseurs) sont surchargés par le traitement des dernières demandes provenant des clients. Autrement dit, les courtiers maintiennent une structure de données de type pile et utilisent la primitive MaDKit *purgeMailBox* qui récupère toujours le dernier message reçu. Les messages utilisant un temporisateur vont être perdus après la terminaison du temps spécifié. Nous avons compté dans cette session 67 messages perdus. En revanche, RT-MTOMAS DASD (figure 4.13) illustre distinctement ce scénario :

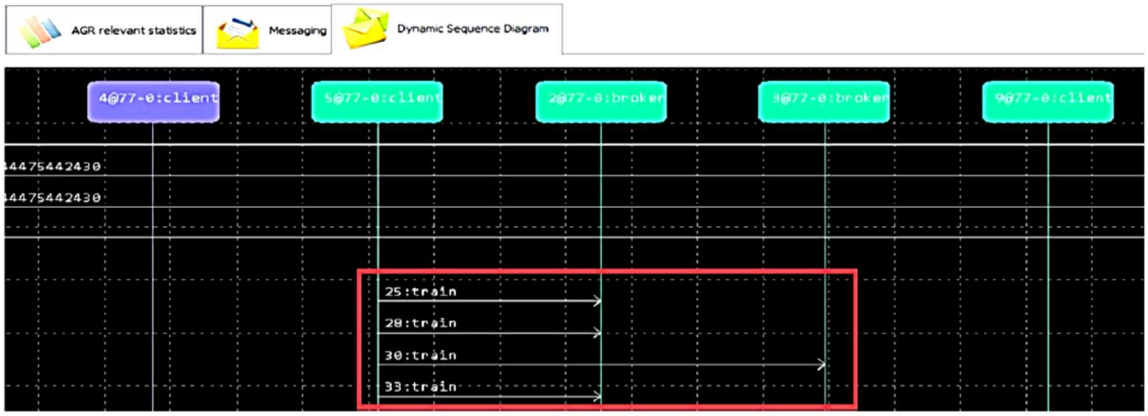


Figure 4.13 : Les messages perdus en raison de la surcharge des courtiers.

En ce qui concerne le modèle AGR, RT-MTOMAS offre un moyen statistique pour traquer les groupes d'agents créés et les différents rôles joués dedans. La figure 4.14 montre que quatre groupes ont été créés (quatre bâtons : 1, 2, 3 et 4 ont remonté) et sont étiquetés respectivement : 201317685073549, 201311884989959, *travel-providers* et *travel-clients* dans lesquels deux groupes ont été créés initialement au démarrage du système (*travel-providers* et *travel-clients*) et les autres groupes étiquetés par un nombre représentent les groupes *contrat* créé par les fournisseurs. L'étiquette du groupe représente le numéro du contrat. Ainsi, on constate que trois groupes ont été créés sans aucune exception générée à l'inverse du quatrième groupe *travel-client* où quatre rôles ont été rejetés (bâton n°5).

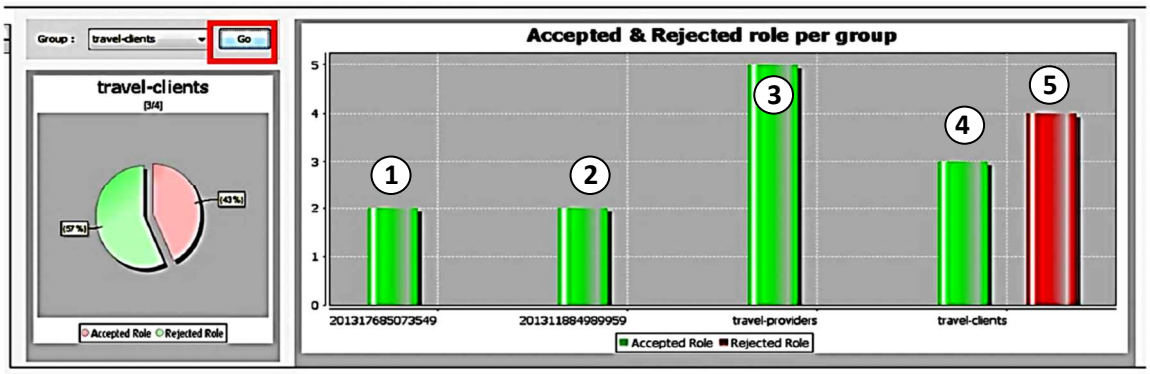


Figure 4.14 : Le nombre de rôles acceptés et rejetés pour chaque groupe créé.

En d'autres termes, la raison avec laquelle des rôles ont été rejetés possède plusieurs arguments à savoir, un agent demande un rôle dans un groupe inexistant ou le même agent demande de jouer le même rôle deux fois. Ainsi, nous avons énuméré dans la section n°3 (table 4.3) les raisons possibles avec lesquels une exception sera générée dans le contexte du modèle AGR.

Une fois la session de monitoring est achevée, un résumé statistique au format XML sera élaboré automatiquement comportant : les actions effectuées par chaque agent (figure 4.15-a), les groupes créés et détruits, les rôles demandés et rejetés dans chaque groupe (figure 4.15-b). etc. Par exemple, dans la figure suivante (figure 4.15-a) le client (@5) a entré dans le système en exécutant la section *activate*, ensuite, il exécute la primitive *createGroupIfAbsent* (figure 4.15-a, ligne : 10) suivie d'une demande de rôle exprimé par la primitive *requestRole* (figure 4.15-a, ligne : 11). Par contre, le client (@10) entre dans la section *living* où il a tenté plusieurs fois d'envoyer un message en utilisant la primitive *sendMessageWithRoleAndWaitForReply* en faisant une pause entre chaque deux tentatives jusqu'à la réception d'une réponse.

```
9  <Agents Name="Client-5 (ACTIVATED)">
10     <Action>Client.createGroupIfAbsent(..)</Action>
11     <Action>Client.requestRole(..)</Action>
12     <Action>Client.pause(..)</Action>
13 </Agents><Agents Name="Client-10 (LIVING)">
14     <Action>Client.sendMessageWithRoleAndWaitForReply(..)</Action>
15     <Action>Client.pause(..)</Action>
16     <Action>Client.sendMessageWithRoleAndWaitForReply(..)</Action>
17     <Action>Client.pause(..)</Action>
18     <Action>Client.sendMessageWithRoleAndWaitForReply(..)</Action>
19     <Action>Client.pause(..)</Action>
20     <Action>Client.sendMessageWithRoleAndWaitForReply(..)</Action>
21     <Action>Client.pause(..)</Action>
22     <Action>Client.sendMessageWithRoleAndWaitForReply(..)</Action>
```

Figure 4.15-a : Un résumé des actions effectuées par le client.

```
</Groups>
<Groups Name="travel-providers">
    <Role>
        <Name>broker: 4@881</Name>
        <State>SUCCESS</State>
    </Role>
    <Role>
        <Name>broker: 3@881</Name>
        <State>SUCCESS</State>
    </Role>
    <Role>
        <Name>boat-provider: 8@881</Name>
        <State>SUCCESS</State>
    </Role>
    <Role>
        <Name>train-provider: 7@881</Name>
        <State>SUCCESS</State>
    </Role>
    <Role>
        <Name>broker: 2@881</Name>
        <State>SUCCESS</State>
    </Role>
</Groups>
<Groups Name="210861730001083">
```

Figure 4.15-b : Les groupes et les rôles demandés pour chaque groupe créé.

Finalement, on peut constater l'intégration des aspects des différents modules de RT-MTOMAS dans le code du SSM. Particulièrement, les trois classes principales des agents client, courtier et fournisseur (figure 4.16). À titre d'exemple, le code de l'aspect *Messaging Aspect* numéroté par 2 et est intégré clairement dans les trois classes des trois agents. De même pour les aspects : *monitoring agent communication* (numéroté par 3), *agent log* (numéroté par 1) et *monitoring agent creation and movement* (numéroté par 4), etc.

Figure 4.16 : La visualisation de l'intégration des aspects dans le code du SSM.

6. Discussion

En observant le cycle de vie de chaque agent ainsi que les interactions échangées et les autres informations pertinentes au modèle AGR, nous constatons clairement l'utilité du langage AspectJ et ses constructions en termes d'instrumentation où des informations variées ont été récoltées. AspectJ offre par le biais des greffons un moyen pour intercepter des points de jonction qui correspondent aux différents

aspects de l'agent à savoir, les actions effectuées, les messages échangés, les groupes créés et les rôles demandés pour chaque groupe.

Un autre résultat surprenant a été découvert en observant l'exécution de cette étude de cas concernant la visualisation du comportement dynamique des agents par RT-MTOMAS DASD. Ce dernier montre, parfois, des lignes de vie des agents sans aucune conversation effectuée (aucun message émis ou reçu). Cela nous mène après avoir examiné profondément le code source de RT-MTOMAS à s'interroger sur la raison d'avoir un agent non communicant dans un SMA ouvert basé-AGR. Ce phénomène figure particulièrement pour le cas des agents courtiers notamment l'agent (@5). Ce dernier joue le rôle de courtier dans les trois groupes : clients, fournisseurs et courtiers. Donc, il communique à la fois avec les clients et les fournisseurs. Un agent (ou une instance d'un agent) non communicant implique que cet agent joue le même rôle dans plusieurs groupes.

Une telle situation nous confirme que ce souci provient de la plateforme MaDKit qui gère la création des groupes. Ainsi, MaDKit engendre plusieurs copies (instances) d'un agent jouant plusieurs rôles. La communication en conséquence s'effectue uniquement avec la dernière instance de l'agent et les autres instances restent figées et ne communiquent plus. Autrement dit, si l'agent courtier demande de jouer un rôle dans le groupe des clients puis il demande de jouer le même rôle dans le groupe des fournisseurs, MaDKit crée deux instances de l'agent courtier, mais l'ensemble des clients et des fournisseurs communiquent uniquement avec l'instance de l'agent courtier appartenant au groupe des fournisseurs et l'instance du courtier appartenant au groupe des clients reste figée. Par conséquent, RT-MTOMAS intercepte chaque primitive *requestRole* validée par MaDKit et dessine la ligne de vie correspondante. La raison avec laquelle les messages s'adressent à la dernière instance créée (dernière ligne de vie). Cela montre un paradoxe avec les spécifications techniques de la version MaDKit 5.0 où l'agent doit communiquer exclusivement avec les autres agents appartenant dans le même groupe (Chebout, Mokhati, Badri , & Babahenini, 2019).

Dans le but de valorisé notre proposition de monitoring des SMA ouverts basés-AGR, nous projetons les caractéristiques de RT-MTOMAS sur les critères élaborés dans l'étude comparative (table 3.3). RT-MTOMAS rejoint la proposition de *AgentSpotter* sur le point de la collecte des informations spécifiques aux SMA (table 4.5). Tandis que, RT-MTOMAS ne prend plus en considération l'évaluation des performances orientées agent à l'inverse de *AgentSpotter.* Ainsi, RT-MTOMAS se focalise principalement sur un type bien déterminé des SMA ouverts où le modèle organisationnel AGR s'est intégré. Pour le critère d'instrumentation, RT-MTOMAS profite amplement des techniques apportées par AspectJ à l'opposé de toutes les autres propositions précédentes qui utilisent l'instrumentation JAVA. Finalement, RT-MTOMAS génère le diagramme de séquence d'agent (RT-MTOMAS DASD) dynamiquement comme étant un modèle de communication en temps réel. RT-

MTOMAS DASD souffre de quelques lacunes et nécessite des améliorations en termes de fragments combinés où l'envoi des messages peut se faire en combinant trois manières possibles : l'envoi parallèle avec l'opérateur logique AND, l'envoi exclusif avec l'opérateur XOR ou l'envoi inclusif avec l'opérateur OR.

CRITÈRES / OUTIL	PLATEFORME AGENT	COLLECTE DES INFORMATIONS SPÉCIFIQUES AUX SMA	ECHANGE DE MESSAGES	ÉVALUATION DES PERFORMANCES DU SMA	ÉVENEMENTS BASES -AGR	MONITORING EN TEMPS REEL	ACTIONS EFFECTUÉES	TECHNIQUE DE PROFILAGE	MODÈLE COMPORTEMENT-ALE
AgentSpotter	**Agent Factory**	√	√	√			√	**Instrument-ation JAVA**	**Graphe d'appel orienté agent**
Esteva et al.	**JADE**		√			√	√		
Mani et al.			√			√			**Diagramme de séquence UML**
Sniffer JADE	**JADE**		√			√		**Instrument-ation JAVA**	**Diagramme de séquence UML**
Zeus toolKit	**ZEUS**		√			√	√	**Instrument-ation JAVA**	**Outil vidéo**
RT-MTOMAS	**MaDKit**	√	√		√	√	√	**Instrument-ation AspectJ**	**RT-MTOMAS DASG**

Table 4.5 : Critères traités par notre proposition.

7. Conclusion

Dans ce chapitre, nous avons présenté notre proposition sur le monitoring des SMA ouverts basés-AGR. Nous avons couronné notre proposition par un outil logiciel baptisé : RT-MTOMAS pour RealTime Monitoring Tool for Open Multi-Agent Systems. Les fonctionnalités de l'outil développé ont été illustrées à travers une étude de cas sous la plateforme MaDKit.

Le monitoring supporté par RT-MTOMAS s'effectue en temps réel ou encore en parallèle avec le SSM. Ainsi, RT-MTOMAS se base principalement sur la technique de monitoring orienté-aspect qui est relativement nouvelle par rapport aux travaux existants. Par ailleurs, les développeurs MaDKit peuvent profiter vivement de RT-MTOMAS pour examiner, analyser et visualiser leurs applications. RT-MTOMAS fournit également un modèle comportemental pour traquer et observer la communication des agents. Le modèle généré est déployé sous le nom RT-MTOMAS DASD pour RT-MTOMAS Dynamic Agent Sequence Diagram. Ainsi, le modèle RT-MTOMAS DASD nécessite des améliorations afin de l'adapter encore plus avec les spécificités des diagrammes de séquences AUML.

Nous allons aborder dans le chapitre suivant notre deuxième proposition pour le contrôle des SMA ouverts basés-AGR.

Chapitre 5

UNE APPROCHE DE CONTRÔLABILITÉ DES SYSTÈMES MULTI-AGENTS OUVERTS BASÈS-AGR

Sommaire

1. Introduction

Dans l'objectif de profiter des résultats offerts par l'approche de monitoring précédemment discutée, une deuxième approche a été proposée dans le but d'influencer, à travers des moyens d'action, le comportement des SMA ouverts de manière à le diriger vers l'état cible souhaité. Les moyens d'action utilisés dans l'approche de contrôle sont exprimés en termes de normes. À cet effet, nous avons contribué dans le domaine de contrôle des SMA par une nouvelle approche baptisée : NorCtrl4OMAS pour : Norms-based ConTRoL for Open Multi-Agent Systems. Une nouvelle approche basée sur les normes pour contrôler le comportement des SMA ouverts basés-AGR. Les normes dans NorCtrl4OMAS sont implémentées en utilisant JESS, un langage de script qui permet l'écriture des systèmes experts. Le choix de JESS est justifié par la possibilité de raisonnement sur les normes afin de sélectionner la norme la plus adéquate au contexte d'exécution. On peut également, créer, modifier ou mettre à jour dynamiquement des normes au fur et à mesure que l'exécution du système progresse. NorCtrl4OMAS est sanctionnée par un outil logiciel et est validée, également, sur une étude de cas sous la plateforme MaDKit.

Dans ce chapitre, nous débutons par présenter notre approche de contrôle proposée. Ensuite, nous illustrons l'outil développé et son applicabilité à travers une étude de cas. Finalement, l'apport de NotCtrl4OMAS par rapport aux propositions existantes présentées dans le deuxième chapitre est bien clarifié à la fin de ce chapitre.

2. Approche proposée

Tel qu'il est indiqué dans le chapitre précédent, notre proposition (Chebout, Mokhati, Badri, & Babahenini, 2016)(Chebout, Mokhati, Badri , & Babahenini, 2019) consiste à diviser le processus de contrôle en deux étapes (figure 4.1) essentielles : le monitoring et le contrôle proprement dit. Le monitoring se procède par l'observation permanente du SSM et, par conséquent, un résumé (un profil) sur le contexte d'exécution sera élaboré. Quant à la deuxième étape, le processus de contrôle prend en argument le résumé contextuel élaboré et agit sur le SSM par le moyen des normes afin d'assurer l'achèvement de l'état cible désiré. De manière générale, l'approche de contrôle proposée est illustrée dans la figure suivante (figure 5.1) :

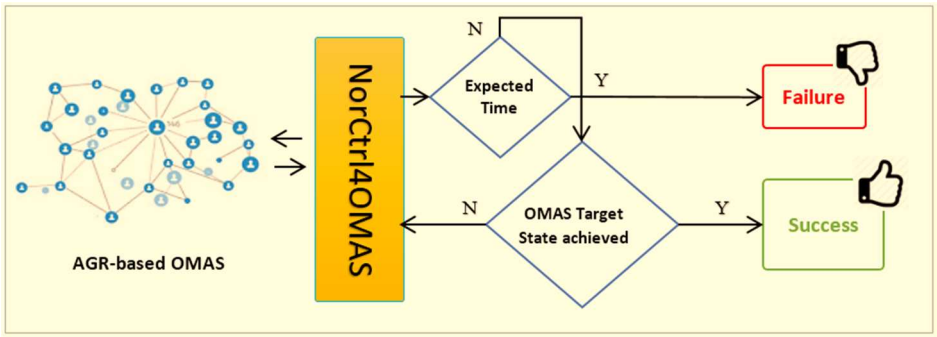


Figure 5.1 : Une vue générale sur NorCtrl4OMAS.

Les normes sont implémentées en utilisant JESS. Dans ce contexte, le choix de JESS est motivé par la capacité avec laquelle on peut raisonner sur les normes dans le but de sélectionner dynamiquement la norme la plus adéquate au contexte d'exécution. Ainsi, JESS utilise les trois types de raisonnement existants : chainage avant, arrière et mixte. La correspondance (matching) sous JESS est basée sur le célèbre algorithme RETE (Forgy, 1982).

La méthodologie de notre approche est présentée dans la figure suivante (figure 5.2). L'idée de notre proposition est de déléguer un SMA particulier pour s'occuper de la tâche de contrôle du SMA ouvert. On parle, donc, d'un SMA orienté-contrôle qui s'exécute en tierce partie entre la plateforme agent et le système contrôlé. Ainsi, le SMA dédié au contrôle est conçu d'une manière distribuée où les agents responsables du contrôle seront dispatchés sur les groupes du système contrôlé. C'est ainsi que la distribution présente de nombreux avantages par rapport à la centralisation en termes de surcharge de communication, consommation de ressources, etc.

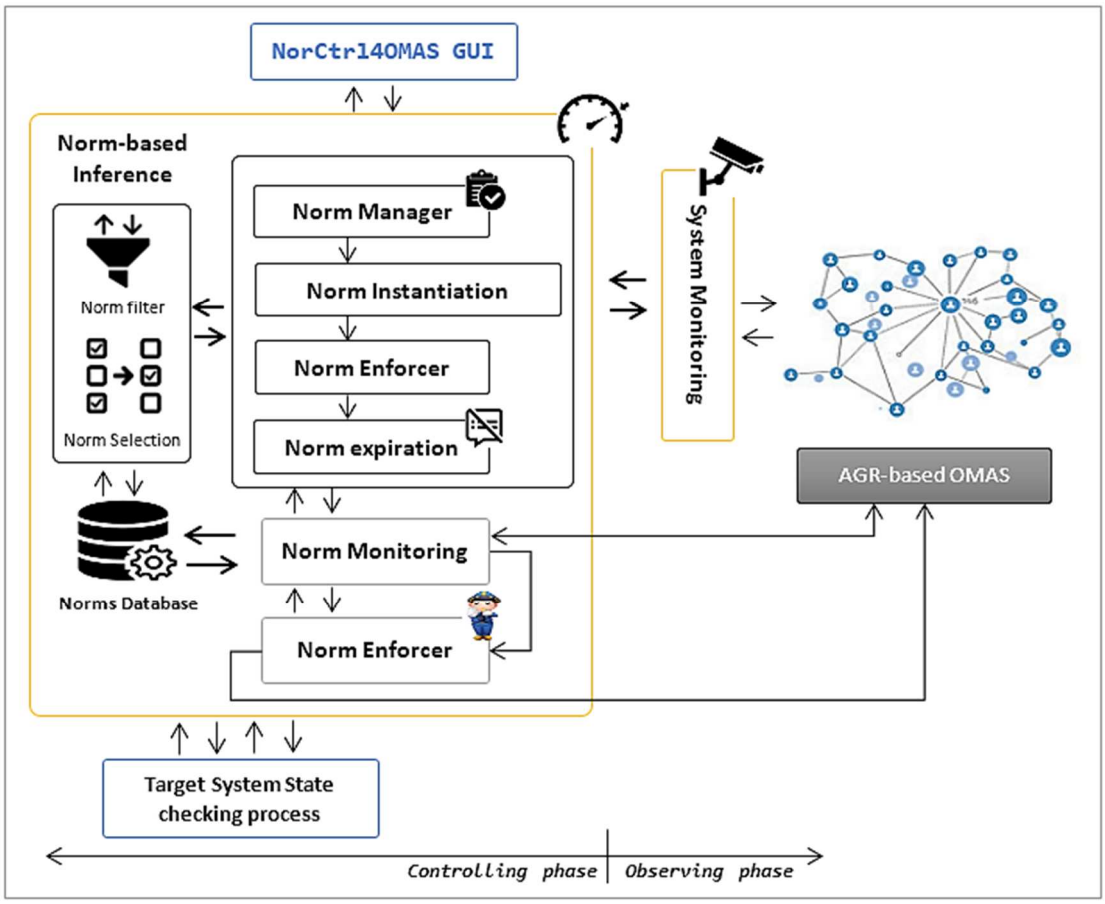


Figure 5.2 : La méthodologie de notre approche.

L'approche proposée est déployée sous le nom : NorCtrl4OMAS pour : **Nor**ms-based **C**on**TR**o**L** for **O**pen **M**ulti-**A**gent **S**ystems. Ainsi, NorCtrl4OMAS est divisé en deux grandes parties (figure 5.2) à savoir : le monitoring et le contrôle proprement dit. Le

monitoring a été réparti, de sa part, en deux sous-parties : le monitoring du système et le monitoring des normes (i.e. monitoring du comportement des agents vis-à-vis celui décrit dans la norme). Les deux types de monitoring sont implémentés sous forme des aspects.

- *La phase de monitoring :*
  Dans le contexte de notre proposition, le moniteur s'exécute en tierce partie (chapitre n°3, section n°6) et consiste en la récolte des informations de type AGR en utilisant un ensemble d'aspects (figure 4.2). Selon le modèle AGR, un agent qui veut entrer ou quitter un groupe doit s'exprimer en utilisant une requête explicite. À titre d'exemple, la plateforme Magentix2 implémente l'entrée d'un agent par le biais de la primitive *aquireRole*, et le départ d'un agent par la primitive *leaveRole*. De même la plateforme MaDKit implémente le départ et l'entrée de l'agent par les primitives *leaveRole* et *requestRole* respectivement.

  La figure suivante (figure 5.3) montre un fragment de code de l'aspect *SystemMonitoring* dans lequel une coupe nommée *observeRequiredRole* (figure 5.3, ligne 1) intercepte tous les appels à la primitive *requestRole*. Ensuite, une liste nommée *activatedRequestedRoleList* (figure 5.3, ligne 10) est maintenue pour la gestion des rôles demandés (ou abandonnés). La liste utilisée a été déclarée sous forme de JAVA Map comportant une clé et une valeur qui correspondent à l'identificateur de l'agent *agent ID* et à la désignation du rôle demandé respectivement.

```
1  pointcut observeRequiredRole(String communityName, String groupName, String roleName, Object passKey) :
2              call(ReturnCode *.requestRole(..)) &&
3              args(communityName, groupName, roleName, passKey);
4
5      after (String communityName, String groupName, String roleName, Object passKey) returning (ReturnCode r):
6              observeRequiredRole(communityName, groupName, roleName, passKey) {
7
8          AbstractAgent agAd = (AbstractAgent) thisJoinPoint.getTarget();
9          if (r.equals(ReturnCode.SUCCESS))
10             activatedRequestedRoleList.put(agAd.getNetworkID(), roleName);
11
12     }
```

Figure 5.3 : Le monitoring des rôles demandés.

  Après avoir récupéré les informations sur le rôle demandé, le moniteur des normes prend en arguments la liste *activatedRequestedRoleList* et cherche dans la base des normes la norme qui correspond au rôle demandé dans le but de la mettre en vigueur.

- *La phase de contrôle :*
  En ce qui concerne le contrôle proprement dit, l'architecture de NorCtrl4OMAS a été conçue de manière distribuée selon le modèle AGR. Autrement dit, chaque agent impliqué dans le processus de contrôle : *NormManager*, *NormInstantiator* ou *NormEnforcer* (figure 5.2) s'intègre au démarrage du système dans tous les

groupes créés. Pour ce faire, chaque agent du processus de contrôle doit exprimer sa demande d'intégration dans un groupe par le biais de la primitive *requestRole* dans tous les groupes qui ont été déjà créés par les agents système. Alors, dans le but de récupérer des informations sur les groupes créés, nous avons implémenté un aspect particulier nommé *GroupCreationTracker* (figure 5.4) qui s'occupe d'intercepter toutes les requêtes de création de groupe par les agents système. Ainsi, l'aspect *GroupCreationTracker* s'exécute en priorité par rapport aux autres aspects en utilisant le mécanisme de précédence offert par AspectJ. À titre d'exemple, sous la plateforme MaDKit, la création d'un groupe se fait par l'une des primitives suivantes, *createGroup* ou *createGroupIfAbsent* (figure 5.4, ligne 6).

```
1  public aspect GroupCreationTracker {
2      int groupID = 0;
3      public static Map<Integer, String> createdGroupList = new HashMap<Integer, String>();
4
5      pointcut observeCreatedGroups(String communityName, String groupName, boolean isDistributed, Gatekeeper keyMaster) :
6              call(boolean *.createGroupIfAbsent(..)) &&
7              args(communityName, groupName, isDistributed, keyMaster);
8
9      after (String communityName, String groupName, boolean isDistributed, Gatekeeper keyMaster) returning (boolean r):
10         observeCreatedGroups(communityName, groupName, isDistributed, keyMaster) {
11         if (r == true) {
12             createdGroupList.put(groupID, groupName);
13             groupID++;
14         }
15     }
16 }
```

Figure 5.4 : L'interception des requêtes de création de groupes.

De même, une liste nommée *createdGroupList* (figure 5.4, ligne 12) a été maintenue pour la gestion des groupes créés. Ainsi, les agents : *NormManager*, *NormInstantiator* et *NormEnforcer* doivent passer des requêtes explicites à chaque élément de la liste *createdGroupList*.

Les principaux acteurs dans le SMA délégué pour le processus de contrôle sont :

- L'agent *Norm Instantiator* : il permet l'instanciation de la norme sélectionnée par le moteur d'inférence. Instancier une norme consiste à créer une copie de cette norme et la substituer par les informations récupérées par l'aspect *observeRequiredRole* : *agent ID* et *requestedRole*. L'instanciation d'une norme donnée est équivalente à sa mise en vigueur. La norme instanciée peut être de trois types différents : *Recommandation*, *Obligation* ou *Interdiction*. Une liste particulière nommée *instantiatedNormList* est maintenue dans le but de gérer les normes instanciées. La désactivation d'une norme (expiration de la norme), en revanche, consiste à la retirer de *instantiatedNormList*. Il est intéressant de mentionner que la manipulation des normes en termes de modification s'effectue toujours sur la copie instanciée de la norme et non sur la norme originale. Ainsi, si un éventuel changement a lieu, il affecte la copie et non la

version originale parce que la modification d'une norme est guidée par le contexte d'exécution qui change à tout moment.

- L'agent *Norm Enforcer* : responsable de contrôler le comportement des agents. Ainsi, il détecte la violation et l'accomplissement des normes et réagit en sanctionnant ou en récompensant les agents.

- L'agent *Norm Manager* : il s'occupe de l'administration des différentes étapes qui correspondent au cycle de vie de la norme à savoir l'instanciation, le renforcement et la désactivation. Ainsi, le *norm manager* prend l'initiative au démarrage du système et observe, à travers les modules *system monitoring* et *norm monitoring*, le contexte d'exécution puis il notifie par un message informatif l'agent responsable de la tâche courante.

Après avoir exposé les différents acteurs impliqués dans le processus de contrôle, il est intéressant de mettre en évidence le type et la structure des normes choisies dans le cadre de notre proposition. Ainsi, nous avons à faire à un type particulier de normes appelé *les normes conditionnelles* où les évènements de déclenchement et d'expiration de la norme sont conditionnés par le temps (deadline). À cet effet, nous avons imposé un ordre temporel entre les trois types de normes sélectionnés (figure 5.5). Le choix des normes conditionnées par le temps s'impose fortement dans notre contexte de recherche où un état cible doit être achevé dans un délai bien déterminé. Donc, les actions et même les messages échangés des agents qui contribuent dans l'achèvement de l'état cible doivent être normalisées. Une action donnée est soumise aux trois types de normes : recommandation, obligation puis interdiction. Si l'agent n'a pas effectué l'action demandée dans la période de recommandation (figure 5.5), il est obligé dans ce cas de la réaliser dans une période additive (période d'obligation) sinon l'action sera interdite d'être accomplie et l'agent sera sanctionné et d'autres mesures seront prises en compte pour récupérer l'action abandonnée.

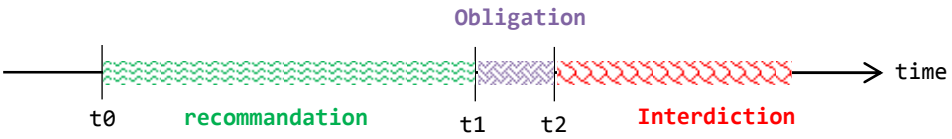


Figure 5.5 : La norme conditionnée par le temps.

Où **:**

t0 : temps de déclenchement de la recommandation (start time).
t1 : période de la recommandation = temps de déclenchement de l'obligation.
t2 : période de l'obligation = temps de déclenchement de l'interdiction.
Prenant une action *a* à effectuer et un moment donné *x* tel que : $x \geq t0$

- a est recommandée si : $t0 < x \leq t1$.
- a est obligée si : $t1 \leq x < t2$.
- a est interdite si : $x > t2$.

Ainsi, nous procédons à une réparation arbitraire de temps entre les concepts déontiques de la manière suivante :

- 75% du temps est dédié à la recommandation: $t1 = (t2 - t0)\frac{3}{4} + t0$,
  Autrement dit, l'agent est permis d'effectuer une action donnée dans une période équivalente à $t1$.
- 25% du temps est consacré à l'obligation (période de l'obligation) $= t2 - t1$.

À cet effet, la modification des normes consiste à changer dynamiquement les contraintes temporelles de la norme dans l'espoir de donner plus de chance aux agents pour qu'ils puissent effectuer leurs actions. Ainsi, augmenter la marge de temps dans laquelle la norme est activée donne l'opportunité à l'agent pour adapter son comportement avec la norme. La modification des contraintes temporelles de la norme a lieu dans le cas où NorCtrl4OMAS confirme l'impossibilité d'atteindre la cible.

En ce qui concerne la représentation des normes dans notre contexte, nous avons choisi JESS comme étant le langage le plus répandu dans la littérature pour la modélisation et le raisonnement sur des normes. Ainsi, une norme sera représentée sous forme d'une règle JESS (figure 5.6). Nous avons augmenté la règle (particulièrement le Left-Hand Side (LHS)) avec des informations de type AGR pour faire allusion à la spécificité du modèle organisationnel avec lequel le SMA ouvert a été implémenté avec. Après avoir récupéré des informations de type AGR (agentID, groupeName et roleName) (figure 5.6, ligne 19), nous procédons à une série de tests pour vérifier l'état actuel de l'agent et le contexte normatif qui correspond à cet état (figure 5.6, ligne 19-24). Si le LHS de la règle correspond au paramétrage actuel (état de l'agent + conditions temporelles), un fait sera inséré dans la base de fait JESS indiquant que l'agent a respecté cette norme.

```
1  (deftemplate AuthorPaperSubmission-Process
2      (slot agentid)
3      (slot group  (type STRING) (default "Authors"))
4      (slot role   (type STRING) (default "Author"))
5      (slot status (type STRING)))
6
7  (deftemplate RelevantDates
8      (slot RegistrationdeadLine  (type LONG))
9      (slot authorRegistrationObligationStarTime (type LONG)))
10
11 (deftemplate AuthorPaperSubmission-RegistrationObligation
12     (slot agentid) (slot role) (slot group) (slot status (type STRING)))
13
14 ; check if this Author is registred
15  (defglobal ?*currentdate* = (System.currentTimeMillis))
16  (defrule pre-registration ""
17     (AuthorPaperSubmission-Process
18         (agentid ?agID) (role ?r) (group ?gr) (status ?s))
19     (RelevantDates
20         (RegistrationdeadLine ?rdl)
21         (authorRegistrationObligationStarTime ?arost))
22     (test (= ?s "waiting"))
23     (test (<= ?*currentdate* ?rdl))
24     (test (> ?*currentdate* ?arost))
25    => (assert (AuthorPaperSubmission-RegistrationObligation
26           (agentid ?agID) (role ?r) (group ?gr) (status "preregistred")))
```

Figure 5.6 : Un exemple d'une obligation conditionnée par le temps.

Concernant la faisabilité de la modification des normes, JESS offre un moyen à travers le constructeur *modify* pour changer la valeur d'un slot particulier en ajoutant ou en soustrayant une quantité donnée.

3. Le processus de renforcement des normes dans NorCtrl4OMAS

Dans le but de bien élucider le rôle de chaque acteur impliqué dans le processus de contrôle (figure 5.2), nous allons dans ce qui suit illustrer à travers un diagramme d'activité UML le flot de contrôle décrivant la tâche de chaque acteur en mettant l'accent sur la norme *obligation* à titre d'exemple.

Tel que mentionné précédement, le module *systemMonitoring* intercepte le rôle demandé en maintenant la liste: *activatedRequestedRoleList* dans laquelle des informations sur le rôle demandé seront enregistrées (figure 5.3, ligne 10). Les rôles demandés seront envoyés en arguments à l'agent *normManager* qui informe par le biais d'un message informatif l'agent *normInstantiator* par le fait d'instancier la norme qui correspond au rôle demandé. À cet effet, une liste a été créée sous le nom : *instantiatedNormList* pour la gestion de normes instanciées. Le choix de la norme adéquate au contexte de l'exécution en cours se fait par l'intermédiaire de l'algorithme RETE incorporé dans JESS. Ainsi, RETE utilise la technique de filtrage par motif (*pattern matching*) pour la sélection d'une norme de la base des normes. Si plusieurs normes ont été sélectionnées pour être instanciées, un mécanisme de filtrage aura lieu. Les techniques de sélection et de filtrage ont été bien clarifiées dans l'algorithme RETE.

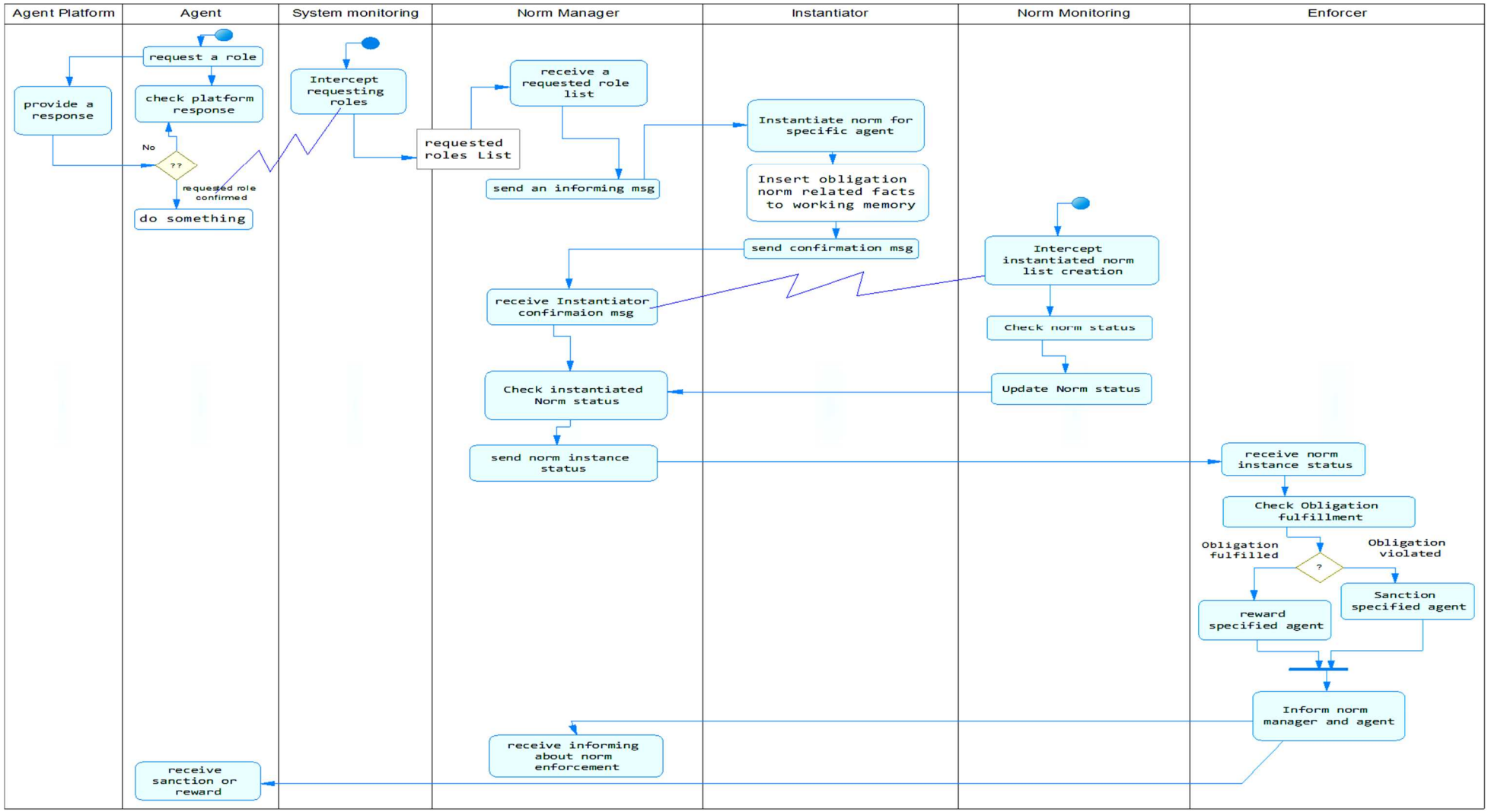


Figure 5.7: Le flot de contrôle pour une norme de type *obligation*.

Le module *normMonitoring* observe d'une manière permanente les changements effectués sur *instantiatedNormList*, et récupère à tout moment la norme en vigueur dans le but de savoir si le comportement de l'agent s'harmonise avec celui décrit dans la norme. Dans le cas d'une obligation, le module *systemMonitoring* intercepte l'action en cours de l'agent, puis il passe en argument l'identificateur de cette action au module *normMonitoring* qui vérifie si l'action effectuée par l'agent conforme avec l'obligation décrite dans la norme instanciée. Autrement dit, est-ce que l'agent respecte l'obligation spécifiée dans la norme pour l'action dont il est en train d'effectuer. La confirmation de l'accomplissement ou la violation de la norme est vérifiée après la date limite (deadline). Si l'agent respecte l'obligation imposée par la norme avant la date spécifiée dans la norme il sera dans ce cas récompensé, sinon il sera puni.

Dans le cas où l'agent respecte la norme, un fait sera ajouté à la mémoire de travail portant des informations de type AGR (identifiant de l'agent, désignation du rôle et le groupe dans lequel le rôle a été accompli). Le processus de punition ou de récompense est entamé par l'agent *enforcer*. Pour cela, une liste des agents punis ou récompensés sera maintenue. Ainsi, *sanctionnedAgentList* et *rewardedAgentList* sont des listes comportant les identifiants des agents sanctionnés ou récompensés respectivement. Dans NorCtrl4OMAS la sanction (ou la récompense) se limite à insérer l'identifiant de l'agent sanctionné à *sanctionnedAgentList* (ou *rewardedAgentList*). Cela a pour objectif de faire une sorte d'arrestation temporaire de l'agent sanctionné et par conséquent les rôles demandés par cet agent ne seront plus accordés prochainement.

Le processus de sanction d'un agent est complètement implémenté par AspectJ. Ainsi, le greffon *around* permet d'orienter complètement le contexte de l'exécution à l'intérieur de l'aspect. En d'autres termes, si un agent sanctionné (son identifiant figure dans *sanctionnedAgentList*) veut demander un rôle, on peut l'interdire facilement en utilisant la primitive *proceed* du greffon *around*. Cela montre l'apport de NorCtrl4OMAS par rapport à d'autres propositions par l'utilisation des techniques de la POA pour la gestion du processus de renforcement des normes. Par ailleurs, le renforcement de la norme de type obligation sera entamé si l'agent n'a pas effectué l'action soumise à une obligation. Tandis que, la sanction dans le cas d'une interdiction aura lieu si l'agent a effectué l'action soumise à une interdiction.

4. Le processus de vérification de l'achèvement de l'état cible souhaité

En ce qui concerne l'état cible du système, un processus de vérification de l'achèvement de l'état cible souhaité a été proposé dans le cadre de NorCtrl4OMAS dans le but d'assurer à un système d'arriver à achever son but. Le processus proposé est normé : TSSCP pour *Target System State Checking Process* (figure 5.8).

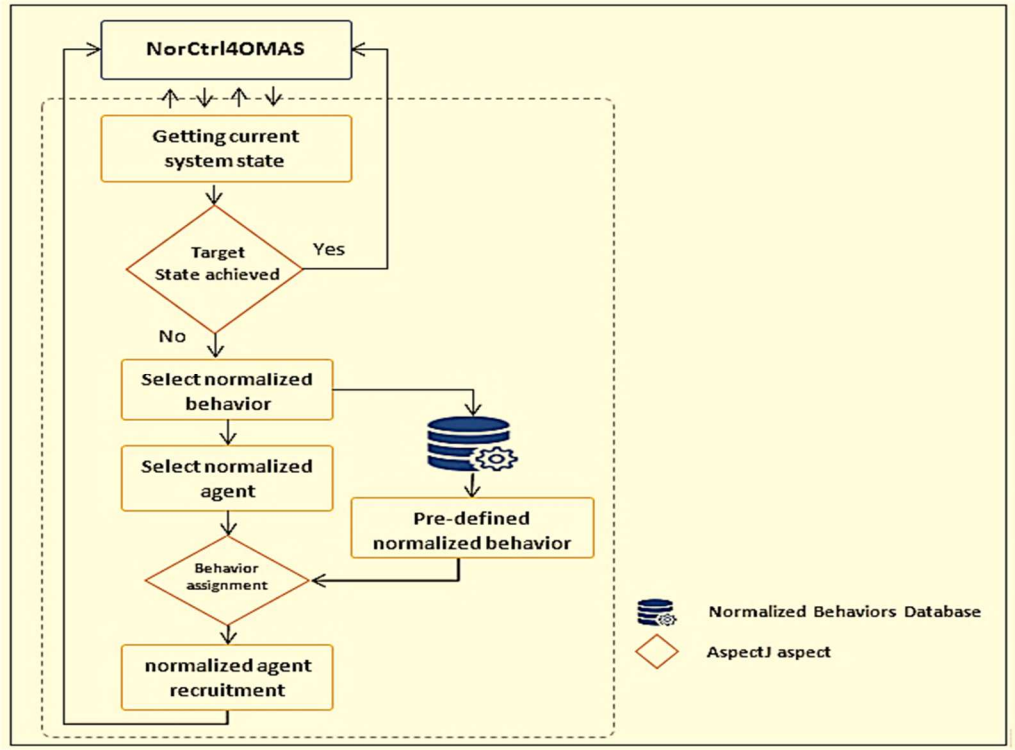


Figure 5.8 : Le processus TSSCP.

Le processus TSSCP consiste à récupérer l'état actuel du système avec toute sa configuration et de le tester avec la configuration désirée. Ainsi, l'état courant d'un système donné dépend fortement du système étudié. Une spécification de l'état souhaité doit avoir lieu avant de commencer le TSSCP. Dans le but de récupérer l'état courant du système, un ensemble d'aspects sont mis en œuvre pour intercepter et traquer les principaux acteurs impliqués dans l'achèvement de l'état cible du système.

Après avoir confirmé l'impossibilité d'atteindre l'état cible du système après plusieurs modifications des contraintes temporelles des normes, le TSSCP procède à une exclusion de tous les agents provoquant cette situation indésirable et recrute, en conséquence, des agents normatifs dont leurs comportements conforment, par supposition, aux normes. De même les agents normatifs sont injectés dans le système en passant par des demandes explicites via la primitive *requestRole* dans tous les groupes créés. Ainsi, le processus d'exclusion consiste à la mise en quarantaine des agents provocateurs et d'interrompre toutes les communications avec eux. Plus de détails sur le processus TSSCP seront expliqués dans la section étude de cas.

5. Outil développé

L'approche NorCtrl4OMAS est supportée par un outil logiciel qui concrétise les différents aspects discutés précédemment. L'outil développé a été validé avec une étude de cas concrète sous la plateforme MaDKit. L'outil NorCtr4OMAS se lance

parallèlement avec le système mis sous contrôle. Nous allons dans ce qui suit montrer l'interface de l'outil NorCtrl4OMAS et ses différentes parties.

Les spécifications techniques de l'outil développé seront présentées dans la table suivante (table 5.1) :

TOOL	USED VERSION
MADKIT VERSION	5.0.5.2 **build-id: 20150430-2053**
ECLIPSE	Juno Service Release 2 **Build id:20130225-0426**
AJDT	AJDT 2.2.3.e42x-RELEASE-20130625-1400 AspectJ version: 1.7.3.20130613144500-a
JESS	Version: 7.1.0
JFREECHART	1.0.13
APPROPRIATE SCREEN RESOLUTION	1366*768

Table 5.1 : Spécifications techniques de l'outil développé.

L'écran n°1 (figure 5.9) montre en temps réel un résumé sur les différentes étapes d'instanciation et de renforcement des normes. Ainsi, des informations pertinentes ont été récoltées comme, *NormID, DeonticLogicOperator, NormDesignationAcronym, NormContext, AgentID, RoleName et NormStatus*. L'état de la norme (*normStatus*) décrit le résultat du module *norm monitoring* en termes d'accomplissement ou violation. Ainsi, l'outil NorCtrl4OMAS traque en temps réel le cycle de vie de chaque norme commençant par son instanciation jusqu'à l'expiration. Des statistiques sous forme de diagramme en bâtons sont illustrées montrant le nombre de normes instanciées. La quantification des normes instanciées est nécessaire pour des mesures de performance afin de mettre en valeur notre proposition par rapport aux contrôleurs existants à l'instar de (Criado, Argente, Noriega, & Botti, 2013).

Par ailleurs, le renforcement des normes a été bien élucidé dans l'outil NorCtrl4OMAS par le biais du deuxième écran de la (figure 5.9) où des diagrammes circulaires montrent en temps réel le taux d'accomplissement et de violation des normes. De plus, la mémoire de travail JESS a été présentée comportant l'ensemble des faits JESS ajoutés après avoir exécuté les règles correspondantes aux différentes normes utilisées.

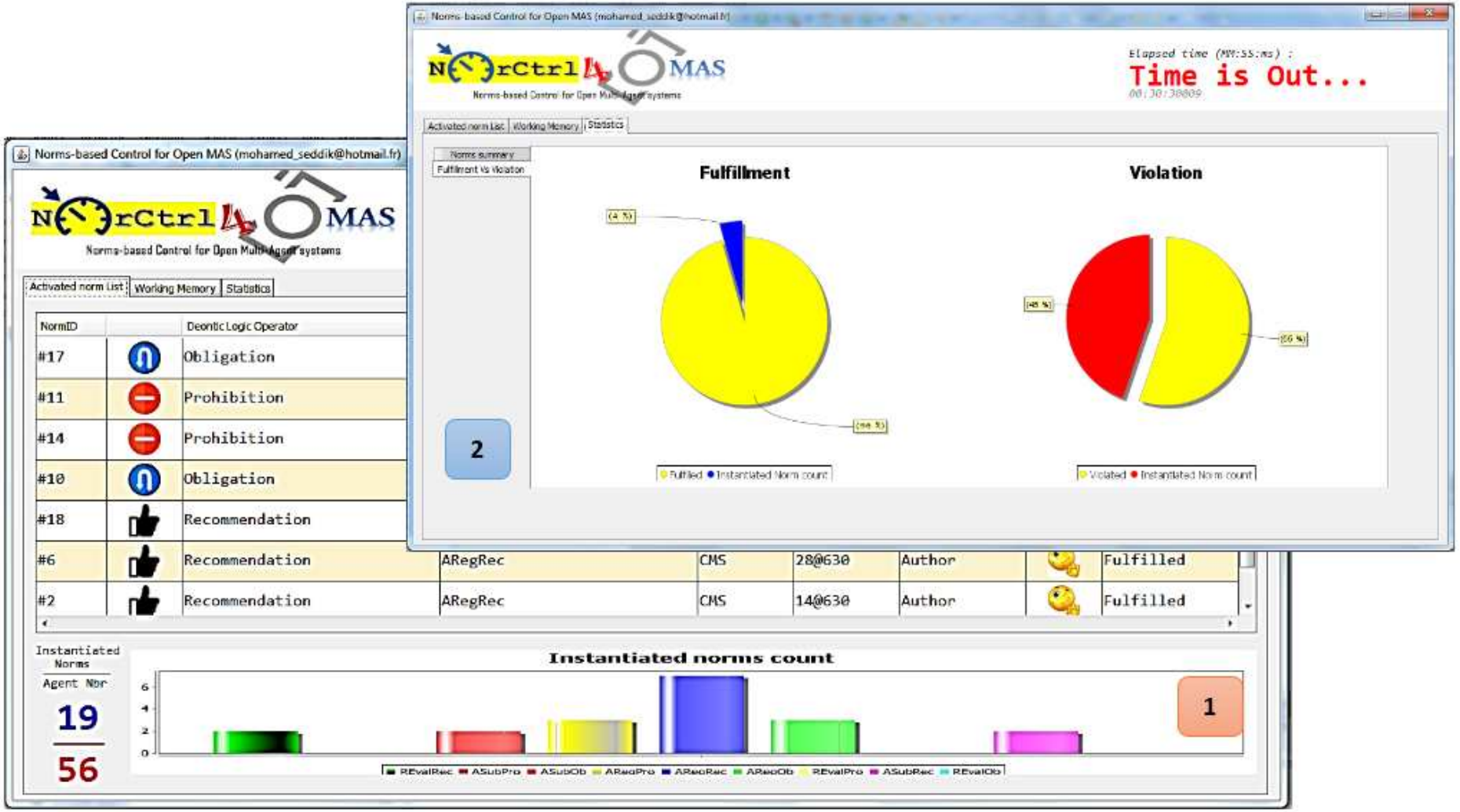


Figure 5.9 : Les captures d'écran de l'outil NorCtrl4OMAS.

6. Étude de cas

Dans l'objectif de valider notre approche, nous avons développé une étude de cas nommée CRS pour *Conference Review System*. Ainsi, les principaux acteurs dans un CRS sont les auteurs, les reviewers et le PC chair. Selon (Schwabe, 2001), le rôle d'un auteur consiste à soumettre un papier à la conférence pour en avoir une acceptation. Le reviewer, en revanche, est responsable pour l'évaluation de l'ensemble des papiers qui lui sont associés et recommande en conséquence la liste des papiers acceptés et rejetés au PC chair. Ce dernier a pour mission la création de la conférence et les différents thèmes en question, détermination du comité de lecture ainsi que les différentes dates nécessaires pour la soumission du reviewing et la notification des auteurs. Finalement, les différentes étapes de déroulement du CRS seront élucidées dans la figure 5.10 :

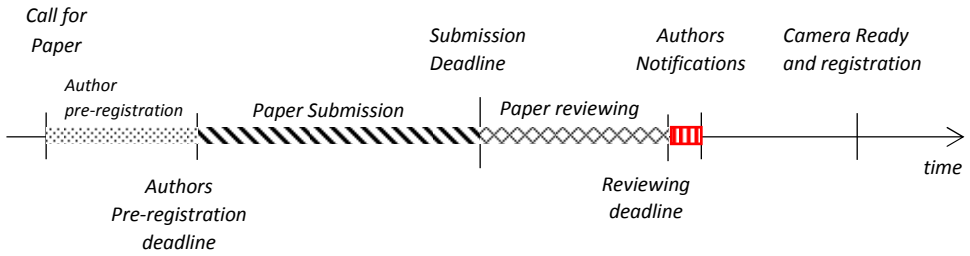


Figure 5.10 : La chronologie des activités du CRS.

Le déroulement normal des différentes étapes du CRS mène à notifier les auteurs dans les délais prédéfinis. Autrement dit, les auteurs auront des notifications sur l'état de leurs papiers avant la date *authorNotificationDeadline*. C'est ainsi que n'importe quel retard qui précède cette date affecte certainement le déroulement ordinaire des différentes tâches. Par conséquent, la période qui sépare le *reviewingDeadline* et *authorNotificationDeadline* représente une période critique.

Dans le cas du CRS, on peut dire que le système atteint son objectif si et seulement si la liste des papiers soumis à la conférence doit être évaluée avant la date *authorNotificationDeadline*. Dans le but de modéliser l'état cible du système, nous avons besoin tout d'abord d'identifier les acteurs qui sont impliqués directement dans l'achèvement de l'état cible du système et les actions pertinentes de chaque acteur. À cet effet, nous avons procédé à la modélisation du comportement de chaque agent en utilisant les diagrammes d'état transition UML (figures 5.11 et 5.12). Dans notre contexte d'étude, nous ne posons aucune supposition sur la capacité nécessaire pour un agent d'accomplir un rôle donné. Ainsi, les agents reviewers, par exemple, disposent de toutes les potentialités nécessaires pour évaluer les papiers qui lui sont assignés.

Pour l'agent Auteur (figure 5.11), nous avons considéré les états *pre-registering* et *paper-submitting*. Un auteur doit être enregistré dans la plateforme de soumission de

papier avant de procéder à une soumission proprement dite. Donc, une relation causale associe les deux états *pre-registering* et *paper-submitting.* Un pré-enregistrement représente dans notre étude de cas un simple envoi du résumé du papier et non le papier intégral. Dans le cas où le résumé est accepté, l'auteur peut envoyer la version intégrale du papier.

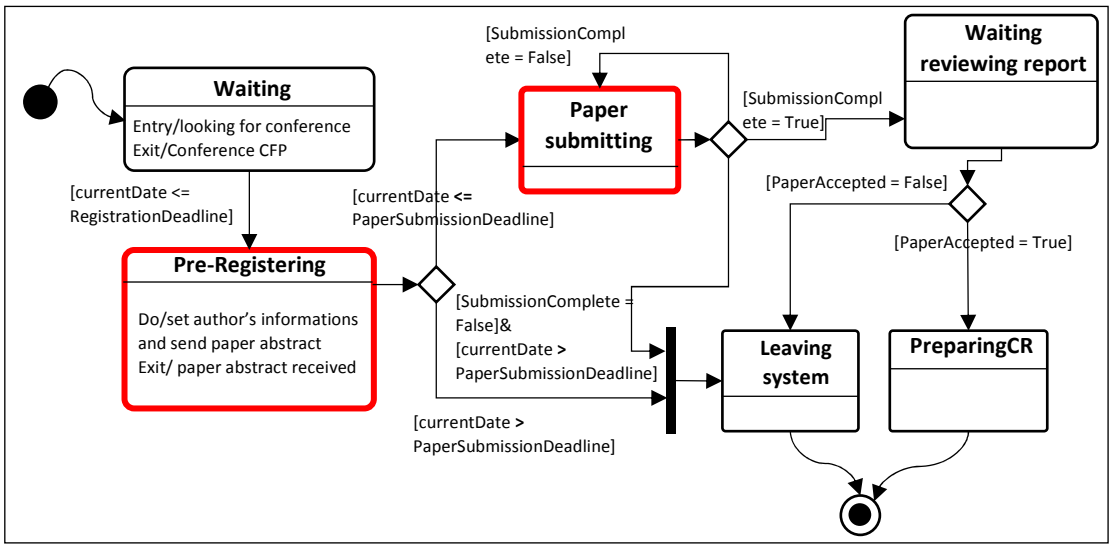


Figure 5.11 : Le diagramme état-transition de l'auteur.

En ce qui concerne l'agent *reviewer* (figure 5.12), on considère uniquement l'état *EvaluatingAssignedPaperList* puisque les reviewers sont généralement enregistrés automatiquement dans la plateforme de soumission par les organisateurs de la conférence. En revanche, n'importe quel retard qui affecte la période d'évaluation des papiers par les reviewers affecte négativement l'achèvement de l'état souhaité du CRS.

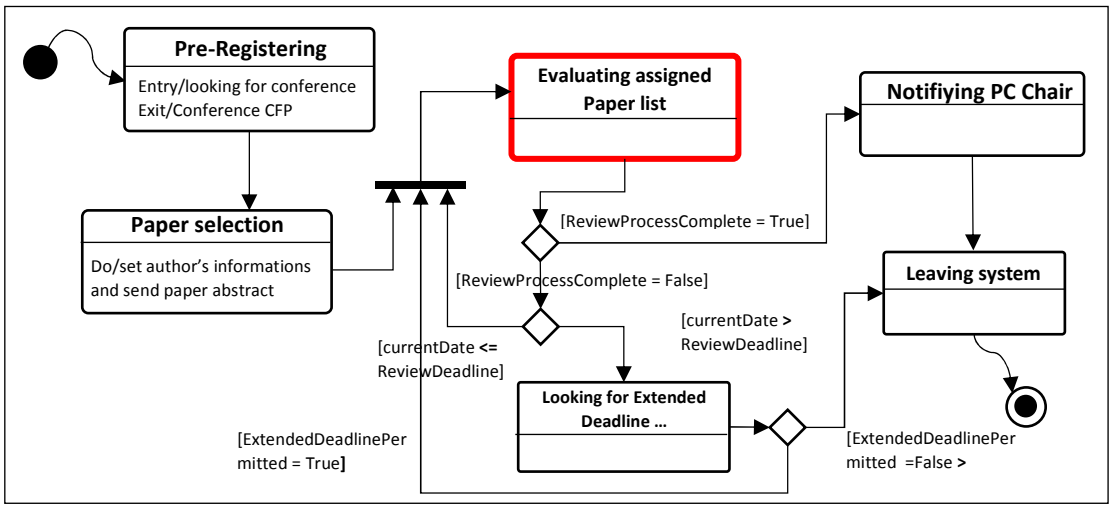


Figure 5.12 : Le diagramme état-transition du reviewer.

Il est intéressant de signaler que les transitions qui séparent les états sont toutes marquées avec des gardes (conditions). Ces dernières représentent un passage

conditionné entre deux états successifs. À titre d'exemple, un auteur ne doit pas soumettre son papier après la date *paperSubmissionDeadLine* (i.e. currentdate <= paperSubmissionDeadline).

Nous avons lancé le CRS avec les deadlines indiqués dans la table suivante (table 5.2) :

Évènement	Deadline	Période
Call for paper	*Au démarrage de système*	
Authors Pre-registration deadline	*après 10 secondes*	*10 secondes*
Submission deadline	*après 17 secondes*	*7 secondes*
Reviewing deadline	*après 27 secondes*	*10 secondes*
Authors Notifications deadline	*après 32 secondes*	*5 secondes*
Camera Ready deadline	*après 37 secondes*	*5 secondes*

Table 5.2 : La configuration initiale du CRS.

En ce qui concerne l'aspect normatif du CRS, nous avons créé un ensemble de normes pour chaque action sélectionnée pour les deux acteurs : auteur et reviewer. La table suivante (table 5.3) montre les différentes normes utilisées et leurs abréviations :

Rôle	Action	Norm	Acronyme	les conditions de la norme (exprimés sous JESS)
Author	**Pre-registration**	**Obligation**	ARegOb	(test (= ?s "waiting")) (test (<= ?*currentdate* ?rdl)) (test (> ?*currentdate* ?arost))
		Prohibition	ARegPro	(test (> ?*currentdate* ?rdl)) (test (= ?s "waiting"))
		Recommendation	ARegRec	(test (= ?s "waiting")) (test (< ?*currentdate* ?arost))
	Paper submission	**Obligation**	ASubOb	(test (= ?s "preregistred")) (test (>= ?*currentdate* ?apsdl)) (test (> ?*currentdate* ?sdl))
		Prohibition	ASubPro	(test (> ?*currentdate* ?sdl)) (test (= ?s "preregistred"))
		Recommendation	ASubRec	(test (= ?s "preregistred")) (test (< ?*currentdate* ?asost))
Reviewer	**Paper Evaluation**	**Obligation**	REvalOb	(test (= ?s "EvaluatingAssignedpaperList")) (test (<= ?*currentdate* ?rdl)) (test (> ?*currentdate* ?rost)
		Prohibition	REvalPro	(test (> ?*currentdate* ?rdl)) (test (= ?s "EvaluatingAssignedpaperList"))
		Recommendation	REvalRec	(test (= ?s "EvaluatingAssignedpaperList")) (test (< ?*currentdate* ?rost))

Legend:
- **s**: agent status.
- **rdl**: pre-registration deadline.
- **arost**: agent registration obligation start time.
- **sdl**: submission deadline.
- **apsdl**: author paper submission obligation start time.
- **asost**: author submission obligation start time.
- **rost**: reviewing obligation start time.
- **rdl**: reviewing deadline.

Table 5.3 : Les différentes normes utilisées dans CRS.

À titre d'exemple, une implémentation sous JESS de la norme ARegOb est montrée dans la figure suivante (figure 5.13) :

```
1  (deftemplate AuthorPaperSubmission-Process
2      (slot agentid)
3      (slot group  (type STRING) (default "Authors"))
4      (slot role   (type STRING) (default "Author"))
5      (slot status (type STRING)))
6
7  (deftemplate RelevantDates
8      (slot RegistrationdeadLine  (type LONG))
9      (slot authorRegistrationObligationStarTime (type LONG)))
10
11 (deftemplate AuthorPaperSubmission-RegistrationObligation
12     (slot agentid) (slot role) (slot group) (slot status (type STRING)))
13
14 ; check if this Author is registred
15  (defglobal ?*currentdate* = (System.currentTimeMillis))
16  (defrule pre-registration ""
17     (AuthorPaperSubmission-Process
18         (agentid ?agID) (role ?r) (group ?gr) (status ?s))
19     (RelevantDates
20         (RegistrationdeadLine ?rdl)
21         (authorRegistrationObligationStarTime ?arost))
22     (test (= ?s "waiting"))
23     (test (<= ?*currentdate* ?rdl))
24     (test (> ?*currentdate* ?arost))
25    => (assert (AuthorPaperSubmission-RegistrationObligation
26           (agentid ?agID) (role ?r) (group ?gr) (status "preregistred")))
```

Figure 5.13 : La norme ARegOb sous JESS.

Afin de savoir quel est le type de la norme qui correspond au contexte d'exécution courant, la figure suivante (figure 5.14) montre une portion de code de l'aspect *CheckingNormType* qui est responsable de vérifier d'une manière permanente la norme la plus appropriée au contexte d'exécution. On peut constater que l'aspect *CheckingNormType* retourne chaque seconde (figure 5.14, line 7) le type adéquat que ce soit : ASubRec, ASubOb, REvalRec ou REvalOb (figure 5.14, ligne 11), etc. tout au long du cycle de vie du CRS.

```
1  public aspect CheckingNormType {
2      public static String normTypeName = null;
3      pointcut observNormTypeOverTime(long t) : call(* *.*longFormat(long)) && args(t) ;
4      @SuppressWarnings("deprecation")
5      before(long t) : observNormTypeOverTime(t) {
6          long currentDate = System.currentTimeMillis();
7          if (currentDate%1000 == 0) {
8          //...
9              if ((currentDate > Enforcer.reviewingObligationStartTime) &&
10                 (currentDate <= Enforcer.reviewingDeadline)) {
11                 normTypeName = NormType.ReviewerObligation.name();
12                 if (!NormMonitoring.currentNormType.contains(NormType.ReviewerObligation.name()))
13                     NormMonitoring.currentNormType.add(NormType.ReviewerObligation.name());
14             }
15             if ((currentDate > Enforcer.reviewingDeadline) &&
16                 (currentDate <= Enforcer.authorNotificationObligationStarttime)) {
17                 normTypeName = NormType.AuthorNotificationRecommendation.name();
18                 if (!NormMonitoring.currentNormType.contains(NormType.AuthorNotificationRecommendation.name()))
19                     NormMonitoring.currentNormType.add(NormType.AuthorNotificationRecommendation.name());
20                 if (!NormMonitoring.currentNormType.contains(NormType.ReviewerProhibition.name()))
21                     NormMonitoring.currentNormType.add(NormType.ReviewerProhibition.name());
22         }
23         // ...
24         NormMonitoring.checkIfNormIsFulfilledOrViolated();
25     }
26 }
27 }
```

Figure 5.14 : La récupération du type de norme durant la chronologie du CRS.

De même, après le deadline d'une norme donnée, l'agent *norm manager* formule des requêtes à la base de normes en utilisant le mot-clé : *defquery* (figure 5.15) dans le but de savoir quel est l'agent qui a été soumis à cette norme. Dans le cas de la norme ARegOb, le paramètre *agentID* (figure 5.15, ligne 10), qui a été récupéré par le module *norm monitoring*, sera passé en argument à la requête afin de récupérer le fait correspondant à la norme ARegOb (particulièrement le RHS). Si c'est bien le cas, la requête retourne des informations de type AGR (*roleName* et *groupName*) correspondant à agentID.

```
1  (deftemplate AuthorPaperSubmission-SubmissionObligation
2      (slot agentid)
3      (slot group)
4      (slot role)
5      (slot status (type STRING)))
6
7  ; get fact list related to author submission obligations
8
9  (defquery getAuthorSubmissionObligationFacts
10   (declare (variables ?agentid))
11   (AuthorPaperSubmission-SubmissionObligation
12         (agentid ?agentid)
13         (group ?gr)
14         (role ?role)
15         (status ?status)))
```

Figure 5.15 : L'extraction de la mémoire de travail des faits en relation avec la norme ARegOb.

En revanche, l'agent *enforcer* utilise un ensemble de règles JESS afin de détecter l'accomplissement ou la violation de la norme en cours. Dans la figure suivante (figure 5.16), la règle *detect-ObligationfulfillmentV2* (figure 5.16, Line 13) se déclenche une fois le LHS de ASubOb (figure 5.16, Line 14) correspondant à l'état courant de l'auteur avec l'ensemble des conditions temporelles indiquées (figure 5.16, lignes 19-21).

Dans ce cas, on peut confirmer l'accomplissement de l'obligation et le fait *paperSubmitting-SubmissionObligationFulfilment* (figure 5.16, line23) sera affirmé.

```
1  (deftemplate paperSubmitting-SubmissionObligationfulfillment
2      (slot agentid)
3      (slot group)
4      (slot role))
5  (deftemplate AuthorPaperSubmission-Obligation
6          (slot agentid)
7          (slot group)
8          (slot role)
9          (slot status (type STRING)))
10 (deftemplate rdS
11     (slot SubmissiondeadLine (type LONG)))
12 (defglobal ?*currentDate* = (System.currentTimeMillis))
13 (defrule detect-ObligationfulfillmentV2
14 (AuthorPaperSubmission-Obligation
15                  (agentid ?author)
16                  (group ?gr)
17                  (role ?r)
18                  (status ?s))
19   (rdS  (SubmissiondeadLine ?sdl))
20   (test (<= ?*currentDate* ?sdl))
21   (test (= ?s "papersubmitting"))
22 =>
23 (assert (paperSubmitting-SubmissionObligationfulfillment
24             (agentid ?author)
25             (group ?gr)
26             (role ?r)))
27 (printout t "NorCtrl4OMAS (Norm Engine) --> Submission Obligation fulfilled for Author " ?author))
```

Figure 5.16: Accomplissement de l'obligation qui correspond à la soumission du papier.

6.1. Scénarios d'exécution du CRS

Deux scénarios possibles peuvent avoir lieu après l'exécution du CRS. Le premier scénario figure dans le cas où le CRS à achevé la cible souhaitée en raison de la conformité du comportement des agents avec les normes créées. Tandis que le deuxième scénario se présente dans le cas où la cible n'a pas été achevée et une intervention du processus TSSCP serait indispensable dans le but de rediriger la configuration courante du système vers la configuration désirée.

Scénario 1 : état cible achevé

Dans ce scénario, tous les agents se comportent conformément à la norme et par conséquent aucun agent n'a été sanctionné que ce soit pour les auteurs ou pour les reviewers.

Dans le même contexte, les agents auteurs et reviewers ont effectué leurs actions dans la majorité des cas dans la période de recommandation et très peu dans la période d'obligation. Ainsi, dans ce scénario sept normes de type recommandation ont été instanciées pour le compte des auteurs qui veulent être pré-enregistrés dans le CRS (figure 5.17, le bâton n° 1). Cela signifie que les auteurs sont intéressés par soumettre leurs papiers dans les bons délais. De même pour l'étape de soumission des papiers. Cependant, quelques auteurs ont été interdits de se pré-enregistrer dans le système. Trois instanciations de la norme ARegPro ont été effectuées dans cet échantillon d'exécution (figure 5.17, le bâton n° 2). Cela est justifié par le fait qu'il y a

des auteurs qui tentent de se pré-enregistrer dans le CRS après la date limite de pré-enregistrement. Concernant l'instanciation de la norme ARegOb où les auteurs ont été pré-enregistrés dans la période d'obligation, cela est justifié par le fait que ces auteurs aient été préoccupés par d'autres tâches ou ils n'ont pas été informés par le CFP (Call For Paper).

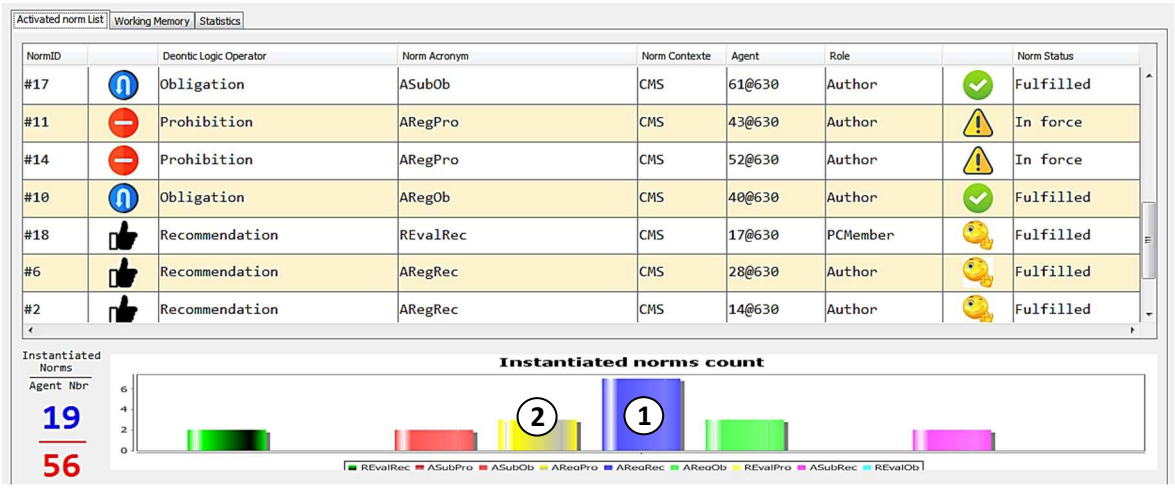


Figure 5.17 : L'instanciation des normes pour le 1er scénario.

Ainsi, le diagramme en bâtons montre qu'il n'y a pas une instanciation pour les normes de type interdiction pour les reviewers (REvalPro). Cela signifie qu'il n'existe plus de papiers qui n'ont pas été évalués par les reviewers.

En ce qui concerne le processus de renforcement des normes, la figure suivante (figure 5.18) montre clairement qu'il n'y a aucune violation des normes (taux de violation égale 0%). Cependant, il y a uniquement 95% des normes qui ont été accomplis. Cela est expliqué par le fait que le nombre de normes instanciées ne correspond pas avec le nombre d'agents. Dans cette session, nous avons énuméré cinquante-six (56) rôles demandés par contre uniquement dix-neuf (19) normes ont été instanciées (figure 5.17) ce qui vaut un taux de 4% (19/56 ~ 4%). Ce taux exprime le nombre de normes à instancier (en attente d'instanciation). Ainsi, les auteurs dont leurs demandes de rôles ont été confirmées et essayent de se pré-enregistrer dans le CRS après la date limite de pré-enregistrement ne sont pas concernés par une éventuelle instanciation de normes pour leurs rôles accordés. De même, les agents qui n'ont pas procédé pour un pré-enregistrement dans le CRS et essayent de se pré-enregistrer après la date limite de soumission ne sont pas concernés également par n'importe quel type de normes parce qu'une relation de précédence associe les deux tâches de pré-enregistrement et de soumission. Nous avons compté trente-sept (37 = 65-19) agents qui ne sont pas concernés par le comportement normatif. Il est intéressant de noter que pendant le développement du CRS le nombre d'agents n'a pas été fixé et est laissé instable afin de concrétiser le caractère ouvert des SMA.

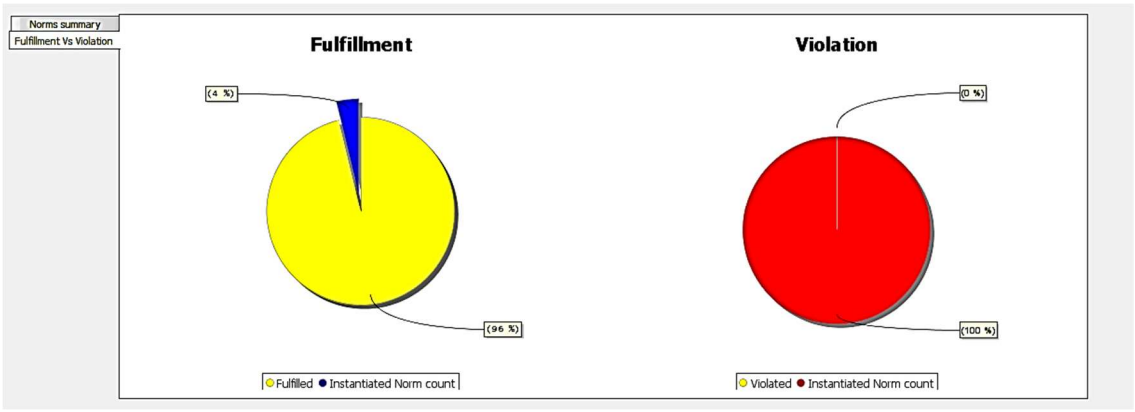


Figure 5.18 : La conformité à la norme pour le 1er scénario.

Senario2: état cible non achevé

Le deuxième scénario correspond au cas où le comportement des agents ne s'adapte pas avec celui normatif. Dans ce cas, une source potentielle de perturbation de système a lieu. Ainsi, on lance le CRS avec une petite modification dans le code source dans la classe de l'agent *reviewer* dans le but de l'occuper pendant un certain moment dans la période d'évaluation des papiers.

La figure suivante (figure 5.19) montre que 65% des normes instanciées ont été violées. Ce pourcentage concerne uniquement les agents reviewers ce qui affecte l'état cible du CRS. Autrement dit, les papiers soumis à l'évaluation n'ont pas été complètement remis avant la date *reviewingDeadline* en raison du retard causé par les reviewers. Tel que cité auparavant, le CRS atteint son objectif si et seulement si tous les auteurs reçoivent leurs notifications avant la date *notificationDeadLine*.

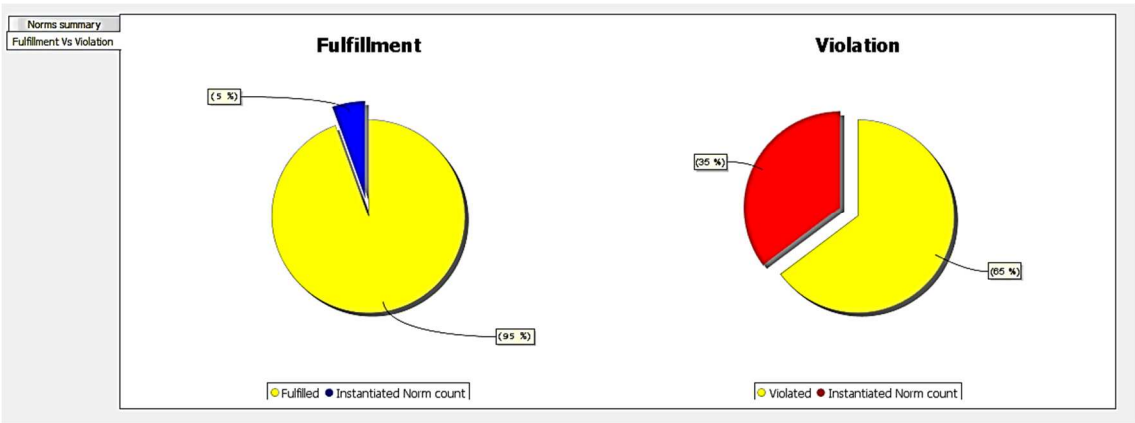


Figure 5.19 : L'accomplissement et la violation de norme pour le 2eme scénario.

Les normes violées sont celles instanciées et n'ont pas été accomplies avant la date d'expiration de la norme. Particulièrement, les normes liées au processus d'évaluation de papiers (REvalRec et REvalOb). Après avoir constaté que la liste des papiers soumis à l'évaluation par les reviewers n'a pas été encore vidée, une mise à

jour des contraintes temporelles de la norme serait indispensable dans le but de ressortir de cette situation et de donner plus de chance aux reviewers pour qu'ils puissent remettre les papiers en question. Ainsi, la modification de la norme consiste à ajouter une quantité de temps à la date *reviewingDeadline*. La figure suivante (figure 5.20) montre le moyen utilisé sous JESS pour récupérer la norme correspondante de la base des normes et procéder à une mise à jour dans le slot Extrdl (*Extended Reviwiewing DeadLine*). Pour ce faire, on récupère l'identifiant du fait (RHS) qui correspond à l'interdiction des reviewers à remettre les papiers (figure 5.20, ligne 10) puis on le modifie (figure 5.20, ligne 20) en rajoutant un délai supplémentaire (5 secondes dans ce cas).

```
1  (deftemplate PcMemberEvaluation-Prohibition
2                (slot agentid)
3                (slot group)
4                (slot role)
5                (slot status (type STRING)))
6  (deftemplate rdRupdate
7       (slot ReviewingDeadline))
8  (defglobal ?*currentDate* = (System.currentTimeMillis))
9  (defrule ExtendedEvaluationDeadline
10     ?p1 <- (PcMemberEvaluation-Prohibition
11               (agentid ?agid)
12               (group ?gr)
13               (role ?role)
14               (status ?st))
15    ?Extrdl <- (rdRupdate
16                (ReviewingDeadline ?rdl))
17    (test (?st "EvaluatingAssignedpaperList"))
18    (test (> ?*currentDate* ?rdl))
19    =>
20    (modify ?Extrdl (+ ?rdl 5000)) ; add 5 secondes
21    (modify ?p1 (?st "EvaluatingAssignedpaperList"))
22    (printout t "NorCtrl4OMAS (Norm Updating) --> Reviewing deadline is extended with : " ?Extrdl crlf))
```

Figure 5.20 : La modification des normes.

Après modification de la norme, si tous les reviewers ne remettent pas encore leurs rapports en temps opportun, le processus TSSCP s'intervient dans le but de faire face à cette situation. Tel que mentionné précédemment, de nouveaux agents normatifs vont être recrutés pour surpasser la situation d'anomalie. Les agents normatifs ont, par supposition, des comportements conforment à la norme et sont implémentés séparément pour des situations pareilles. Ainsi, la violation des normes pour les agents normatifs n'est plus possible.

La figure suivante (figure 5.21) montre que l'intervention des agents normatifs a résolu le problème et les rapports d'évaluation ont été retournés immédiatement. Ainsi, l'agent *norm manager* affecte la tâche d'évaluation des papiers en attente à l'agent *normalizedPCMember* portant l'identifiant (11@621-0) tel qu'illustré par la flèche n°1 dans la figure suivante (figure 5.21). L'agent *normalizedPCMember* renvoie le rapport immédiatement après évaluation (mentionné avec la flèche n°2 dans la figure 5.17). Par conséquent, l'agent *norm manager*, après avoir reçu la réponse de la part de l'agent *normalizedPCMember*, envoie le rapport d'évaluation aux agents organisateurs du CRS (mentionné avec la flèche n° 3 dans la figure 5.17). Ensuite, les

agents organisateurs informent les auteurs par les résultats d'évaluation de leurs papiers (mentionné avec la flèche n°4 dans la figure 5.17).

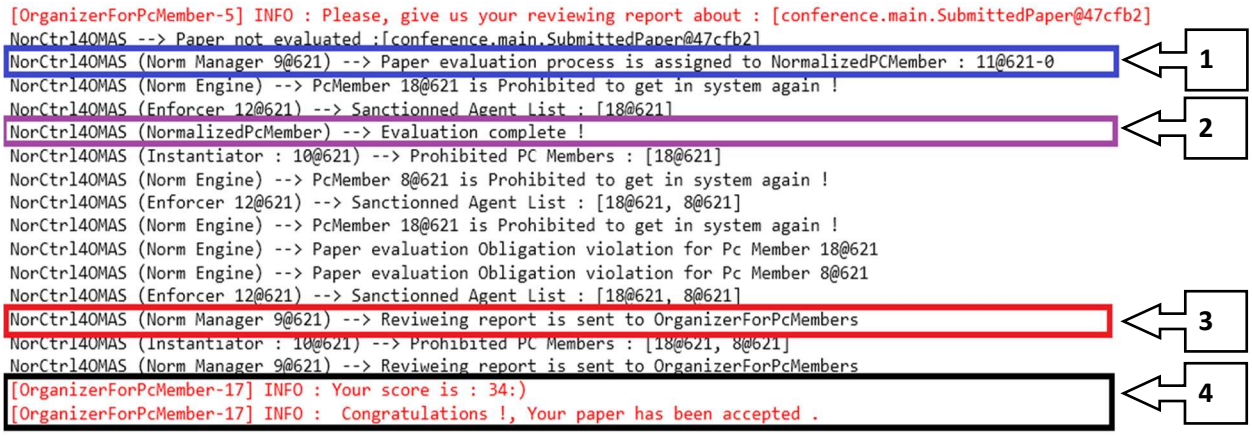


Figure 5.21 : L'intervention des agents normatifs du TSSCP.

7. Discussion

Malgré l'intégration des normes dans le CRS, l'état global attendu n'a pas été achevé après plusieurs tentatives d'exécution. Ainsi, les contraintes comportementales imposées par les normes ne permettent pas à CRS à atteindre son objectif. Cela est justifié par le fait que certains agents ont maintenu leurs autonomies et n'ont pas accepté de se conformer avec les normes. C'est le cas des reviewers qui n'ont pas obéi aux obligations des contraintes temporelles qui correspondent à la date limite de remise des papiers évalués. Par conséquent, la contrôlabilité n'a pas été assurée.

De plus, NorCtrl4OMAS utilise le processus TSSCP pour assurer la contrôlabilité du CRS. Ainsi, TSSCP se base sur les normes comme étant un mécanisme de contrôle. Idéalement, un processus de contrôle sera adressé aux systèmes non normatifs dans le but de guider leurs comportements de manière appropriée. Le processus TSSCP a été appliqué sur un système normatif (CRS classique enrichi avec les normes) dans l'objectif d'assurer sa contrôlabilité, cependant ce n'est plus le cas.

La représentation des normes dans NorCtrl4OMAS a été faite en utilisant le système à base de règles JESS. L'adoption de JESS a été justifiée principalement par sa flexibilité en le basculant avec JAVA facilement. Ainsi, JESS offre un bon support pour exprimer les contraintes temporelles incorporées dans les obligations, interdictions et recommandations. Dans la version normalisée de CRS, des implémentations JESS ont été ajoutées sous forme de paquetages notamment : *normEngine*, *normUpdating*, *normQueries*, *normEnforcement*. Chaque paquetage englobe un ensemble de fichiers JESS liés à chaque agent impliqué dans l'achèvement de l'état souhaité du CRS (les auteurs, et les reviewers). En ce qui concerne la modification des normes, JESS offre la possibilité de mettre à jours quelques slots qui correspondent à la partie LHS de la

règle en procédant par l'extraction de la règle de l'agenda et la modification du slot en question en utilisant le constructeur *modify*.

L'utilisation de la POA a pour objectif l'amélioration de performance liée à la charge de communication imposée par les agents impliqués dans le processus de contrôle dans les approches classiques. En revanche, l'utilisation de la POA nécessite de la prudence dans la manipulation des structures de données partageables. Pour cela, il est préférable d'utiliser les structures concurrentes plutôt que des structures simples. Nous citons à titre d'exemple : ConcurrentHashMap au lieu de HashMap et CopyOnWriteArrayList au lieu de ArrayList cela évitera le déclenchement des exceptions de type ConcurrentModificationException. Finalement, l'utilisation des aspects réduit considérablement le nombre de messages échangés et libère en conséquence les canaux de communications.

Dans CRS, uniquement deux agents ont été soumis à un comportement normatif, les auteurs et les reviewers. Ces agents disposent chacun d'eux, selon leurs modèles état-transition, six états (figures 5.11 et 5.12). Chaque état a été soumis à trois types de normes, recommandation, obligation et interdiction (table 5.4). De plus, chaque norme sera renforcée par le moyen d'une sanction ou récompense (les sanctions dans le cas de violation de la recommandation n'a pas été prise en compte). Nous résumons douze (12) normes créées (la dimension de la base de normes) et soixante (60) manipulations entre NorCtrl4OMAS et le moteur de normes. Dans le cas de CRS normalisé, uniquement 2 actions parmi 6 pour les auteurs et une seule action parmi 6 pour les reviewers ont été maintenues (en somme 3 actions ont été considérées dans l'achèvement de l'état global souhaité). Cela signifie que dix (10 = 2*3*2-2) manipulations ont été effectuées pour le compte de l'agent auteur sans compter le renforcement de la recommandation et cinq (5=1*3*2-1) manipulations ont été effectuées pour le compte de l'agent reviewer sans compter également le renforcement de la recommandation (table 5.4). Ceci indique que le nombre de manipulations de normes augmente considérablement si le nombre d'actions sélectionné pour l'achèvement de but global souhaité s'incrémente. Ainsi, une gestion de la création et l'instanciation des normes doit avoir lieu dans le but d'améliorer les performances de NorCtrl4OMAS par rapport aux contrôleurs existants.

Nom du rôle	Nombre d'actions		Nombre d'actions par normes	Renforcement par norme	Nombre de norme	
	Total	Actions considérées			Total	Actions considérées
Auteur	6	2	3	2	36 – 6 = 30	12-2 = 10
Reviewer	6	1	3	2	36 – 6 = 30	6-1 = 5
				Manipulations	60	15

Table 5.4 : Cardinalité de manipulations de normes.

La table suivante (table 5.5) montre l'apport de NorCtrl4OMAS par rapport aux travaux de l'état de l'art discutés dans le deuxième chapitre. Il est clair que

NorCtrl4OMAS ramène des nouveautés en termes de critères étudiés tels que la modification des normes qui a été exclue dans tous les travaux discutés. Ainsi, NorCtr4OMAS se focalise sur deux aspects contextuels de la société d'agent à savoir l'interaction et l'organisation. Cette dernière est exprimée par le modèle AGR. Tandis que le critère de l'environnement n'a pas été agité et également le critère de résolution de conflits entre les normes. Nous avons supposé au départ que les critères de l'émergence, de transmission et de conflits n'ont pas été pris en charge dans le cadre de NorCtrl4OMAS.

Jusqu'ici nous avons élucidé les différents aspects traités par l'apport de NorCtrl4OMAS, par rapport aux travaux existants, par divers concepts notamment l'étude de la normativité du modèle organisationnel AGR. Néanmoins, notre proposition souffre de quelques lacunes en ce qui concerne les types de normes traitées où un ordre de précédence a été imposé. Autrement dit, une action sera recommandée puis elle rentre dans l'étape d'obligation ensuite elle finira par être interdite pour une durée bien déterminée pour les trois modalités. L'ordre imposé a lieu dans certains contextes particuliers où la validité de la norme a été conditionnée par des dates limites. Ainsi, le critère temps a été amplement adopté dans cette thèse parce que l'achèvement de l'état global du système nécessite sa prise en compte sinon on tombera dans une infinité de manipulations durables.

Un autre point faible de notre approche consiste en la supposition que les agents normalisés recrutés par le processus TSSCP obéissent au comportement normatif sans qu'ils soient soumis au monitoring. En fait, la confiance donnée aux agents normalisés de TSSCP doit être prouvée.

Critères / Proposition	Renforcement des normes		Modification des Normes	Architecture distribuée de renforcement	Actions effectuées	Cycle de vie de la norme	Échange de messages	Représentation des normes	Concepts déontiques				Contexte			Résolution de conflits
	Sanction	Récompense							Permission	Obligation	Interdiction	Recommandation	Interaction	Environnement	Organisation	
Dastani et al.					√			√		√						√
García-Camino et al.	√	√			√			√	√	√	√				√	
Felicíssimo et al.	√	√			√			√	√	√	√		√	√	√	
Figueiredo et al.	√	√			√	√		√	√	√	√			√	√	√
Ahmed et al.	√	√						√		√	√	√				√
Criado et al.	√	√		√	√	√	√	√	√	√	√			√	√	
Mahmoud et al.	√	√				√						√				
Alechina et al.	√	√		√	√			√								
José Plácido et al.	√	√			√	√		√	√	√	√			√		
Marir et al.	√	√			√			√	√	√	√	√				
NorCtrl4OMAS	√	√	√	√	√	√		√	√	√	√	√	√		√	

Table 5.5 : Critères traités par NorCtrl4OMAS.

8. Conclusion

Dans ce chapitre, nous avons proposé une extension de notre approche de monitoring par une proposition de contrôle des SMA ouverts basés-AGR en utilisant les normes. De même, l'approche proposée profite amplement des techniques de la POA pour un double intérêt : le monitoring du système et le monitoring des normes à la fois. Ainsi, notre proposition porte la désignation : NorCtrl4OMAS pour : Norms-based ConTRoL for Open Multi-Agents Systems. Les normes dans NorCtrl4OMAS ont été implémentées sous JESS. Le choix de JESS est motivé par le fait de raisonnement sur les normes dans le but de sélectionner dynamiquement la norme la plus adéquate au contexte d'exécution en cours. Ainsi, JESS utilise le célèbre algorithme RETE pour faire la correspondance d'un fait ou un ensemble de faits à une règle donnée. Enfin, NorCtrl4OMAS a été supporté par un outil logiciel qui est validé, sous la plateforme MaDKit, sur une étude de cas bien dédiée.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

1. Bilan

Notre projet de thèse se place dans le contexte du génie logiciel orienté-agent. Particulièrement nous nous sommes intéressés au problème de la contrôlabilité des SMA ouverts. En fait, la caractéristique de l'ouverture permet à un SMA d'être extensible et évolutif dans le sens où on peut ajouter et/ou retirer dynamiquement des agents pour répondre aux exigences fonctionnelles nouvellement rencontrées. Cependant, la capacité d'insérer (ou de retirer) des agents peut affecter d'une manière inappropriée le comportement du système en question et, par conséquent, des situations indésirables émergent.

Afin de maitriser le comportement d'un SMA ouvert et de le diriger vers un état cible souhaité, des contributions bénéfiques ont été proposées. Nous avons commencé notre thèse par un état de l'art dispatché sur les trois premiers chapitres.

Le premier chapitre porte sur les SMA dans lequel des concepts et des notions inhérentes aux SMA ouverts ont été présentés dans le but de bien clarifier et exposer les limites et les enjeux de ce type de systèmes. Ainsi, le non-déterminisme, l'hétérogénéité, l'émergence et la non-linéarité des systèmes ouverts rendent les tâches de monitoring et de contrôle incontournables.

En revanche, dans le deuxième chapitre nous avons présenté des concepts relatifs à la normativité des SMA. Les normes, les mécanismes de renforcement des normes ainsi que les différentes architectures normatives ont été élucidés.

Dans le troisième chapitre nous avons montré des concepts clés autour le monitoring des SMA et les différentes techniques et outils utilisés dans la littérature notamment

les techniques de la POA. Ainsi, nous avons étudié les techniques de profilage et d'instrumentation en utilisant AspectJ pour les SMA.

Dans le quatrième chapitre, nous avons présenté notre première contribution sur le monitoring des SMA ouverts basés-AGR. Une nouvelle approche de monitoring a été introduite dont l'idée principale est de s'appuyer sur les techniques de la POA pour l'instrumentation et le monitoring du système. En fait, la POA a suscité un grand intérêt dans la littérature surtout en ce qui concerne l'analyse dynamique de logiciels. L'originalité de notre approche de monitoring vient de la prise en compte des spécificités des SMA ouverts basés-AGR par un paradigme de programmation relativement nouveau. Nous avons montré le fonctionnement de notre approche par la couronner par un outil logiciel baptisé : RT-MTOMAS pour RealTime Monitoring Tool for Open Multi-Agent Systems. L'outil développé a été illustré à travers une étude de cas bien concrète sous la plateforme MaDKit.

Par la suite, une extension de notre approche de monitoring pour le contrôle des SMA ouverts basés-AGR a été discutée dans le cinquième chapitre. De même, l'approche proposée pour le contrôle des SMA ouverts profite amplement des techniques de la POA pour un double intérêt : le monitoring du système et le monitoring des normes. Ainsi, notre proposition est déployée sous le nom : NorCtrl4OMAS pour : Norms-based ConTRoL for Open Multi-Agents Systems. Les normes dans NorCtrl4OMAS sont implémentées sous JESS. Le choix de JESS est motivé par le raisonnement sur les normes dans le but de sélectionner dynamiquement la norme la plus adéquate au contexte d'exécution en cours. Enfin, la proposition NorCtrl4OMAS a été supportée par un outil logiciel qui a été validé, sous la plateforme MaDKit, sur une étude de cas bien dédié.

2. Perspectives

Certainement, nos contributions restent ouvertes et extensibles sur plusieurs directions. Nous envisageons la poursuite du travail de monitoring selon plusieurs aspects à savoir :

- Doté RT-MTOMAS par des mécanismes d'évaluation de performances orientées agents : cela vient du fait que le monitoring est effectué en temps réel ce qui affecte considérablement les performances du système en termes de consommation de ressources : CPU, mémoire, canaux de communications, etc.
- Le problème de l'utilisation de ressources mène à des situations critiques dans le cas ou deux ou plusieurs agents se mettent dans une situation d'inter-blocage. Nous pensons qu'il est intéressant d'étendre RT-MTOMAS par des moyens afin de surpasser une telle situation.
- Augmenter le diagramme RT-MTOMAS DASD par les fragments combinés AUML : AND, OR et XOR.

En ce qui concerne l'approche de contrôle proposée, nous prévoyons à court et à moyen terme :

- La proposition d'un protocole d'affectation de rôle basé-AGR, qui aura pour but principal l'assurance *préventive* de la contrôlabilité.
- L'incorporation des normes dans le modèle organisationnel AGR par la proposition d'un méta-modèle AGR normatif.

Bibliographie

Ahmad, A. (2012). An agent-based framework incorporting rules, norms and emotions (OP-RND-E). *Thèse de doctorat de l'université de Universiti Tenaga, Selangor, Malaysia*.

Ahmad, A., Zaliman, M., Yusof, M., Ahmad, M., Ahmed, M., & Mustapha, A. (2011). Resolving conflicts between personal and normative goals in normative agent systems. *7th International Conference on Information Technology in Asia: Emerging Convergences and Singularity of Forms (CITA'11),*.

Alberola, J., Such, J., Botti, V., Espinosa, A., & Garcia-Fornes, A. (2013). A Scalable Multiagent Platform for Large Systems. *Computer Science and Information Systems*, 51-77.

Alberti, M., Gomes, A., Gonçalves, R., Leite, J., & Slota, M. (2011). Normative systems represented as hybrid knowledge bases. *Proceedings of the 12th international conference on Computational logic in multi-agent systems*, 330-346.

Alechina, N., Halpern, J., Kash, I., & Logan, B. (2018). Incentive-Compatible Mechanisms for Norm Monitoring in Open Multi-Agent Systems. *International Joint Conference on Artificial Intelligence*, 5543-5547.

Arcos, J., Esteva, M., Noriega, P., Rodríguez-Aguilar, J., & Sierra, C. (2005). An Integrated Development Environment for Electronic Institutions. *In: Unland R., Calisti M., Klusch M. (eds) Software Agent-Based Applications, Platforms and Development Kits. Whitestein Series in Software Agent Technologies. Birkhäuser Base*, 121-142.

Argente, E., Botti, V., Carrascosa, C., Giret, A., Julian, V., & Rebollo, M. (2011). An abstract architecture for virtual organizations: The THOMAS approach. *Knowledge and Information Systems*, 379-403.

Artikis, A. (2012). Dynamic specification of open agent systems. *Journal of Logic and Computation, 22*(6), 1301-1334. doi:10.1093/logcom/exr018

Artikis, A., & Pitt, J. (2001). A formal model of open agent societies. *Proceedings of the fifth international conference on Autonomous agents* (pp. 192-193). Montreal, Quebec, Canada.

Artikis, A., Sergot, M., Pitt, J., Busquets, D., & Riveret, R. (2016). Specifying and Executing Open Multi-agent Systems. *In: Aldewereld H., Boissier O., Dignum V., Noriega P., Padget J. (eds) Social Coordination Frameworks for Social Technical Systems. Law, Governance and Technology Series, 30*, 197-212. doi:https://doi.org/10.1007/978-3-319-33570-4_10

Avgustinov, P., Bodden, E., Hajiyev, E., Hendren, L., Lhoták, O., de Moor, O., . . . Verbaere, M. (2006). Aspects for trace monitoring. *In: First Combined International Workshops on Formal Approaches to Software Testing and Runtime Verification (FATES/RV)*, 20–39.

Ball, T. (1999). The concept of dynamic analysis. *Proceedings of the 7th European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-7). Springer*, 216–234.

Bates, J. (1994). The role of emotion in believable agents. *Communications of the ACM*, 122-125 .

Bates, J., Bryan Loyall, A., & Scott Reilly, W. (1994). An architecture for action, emotion, and social behavior. *In: Castelfranchi C., Werner E. (eds) Artificial Social Systems. MAAMAW 1992. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*, 55 - 68.

Bellifemine, F., Poggi, A., & Rimassa, G. (2001). Developing Multi-agent Systems with JADE. *In: Castelfranchi C., Lespérance Y. (eds) Intelligent Agents VII Agent Theories Architectures and Languages. ATAL 2000. Lecture Notes in Computer Science, 1986*, 89-103.

Binder, W., Hulaas, J., & Moret , P. (2007). Advanced Java bytecode instrumentation. *Proceedings of the 5th international symposium on Principles and practice of programming in Java*, 135 - 144.

Bodden, E., & Havelund , K. (2008). Effective race detection using AspectJ. *In: Proceedings of the 2008 international symposium on Software testing and analysis (ISSTA)*, 155–166.

Boella, G., & Torre, L. (2004). Regulative and Constitutive Norms in Normative Multiagent Systems. *Proceedings of The 9th International Conference on the Principles of Knowledge Representation and Reasoning*, 255 - 266.

Boella, G., & Torre, L. (2008). Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic*, 152-171.

Boella, G., & Van der Torre, L. (2003). Norm governed multi agent systems : the delegation of control to autonomous agents. *In : Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT)*., 329–335.

Boella, G., & van der Torre, L. (2008). Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic*, 152-171.

Boella, G., Torre, L., & Verhagen, H. (2006). Introduction to Normative Multiagent Systems. *Computational & Mathematical Organization Theory*, 71 - 79.

Boissier, O., Gitton, S., & Glize, P. (2004). Caractéristiques des Systèmes et des Applications. *Systèmes Multi-Agents, Observatoire Français des Techniques Avancées, ARAGO 29*, 25-54.

Bourlès, R., & Henriet, D. (2017). Théorie des jeux. *Support de cours de l'Ecole Centrale de Marseille*.

Broersen, J., Dastani, M., & Torre, L. (2001). Resolving Conflicts between Beliefs, Obligations, Intentions, and Desires. *2143*, pp. 568–579. Springer, Berlin, Germany.

Caire, P. (2007). A Normative Multi-Agent Systems Approach to the Use of Conviviality for Digital Cities. *International Conference on Coordination, Organizations, Institutions, and Norms in Agent Systems III (COIN '07),*, 245-260.

Caldas, J., & Coelho, H. (1999). The Origin of Institutions: Socio-Economic Processes, Choice, Norms and Conventions. *Journal of Artificial Societies and Social Simulation*, 1-1.

Caldas, J., & Coelho, H. (1999). The Origin of Institutions: Socio-Economic Processes, Choice, Norms and Conventions. *Journal of Artificial Societies and Social Simulation, Journal of Artificial Societies and Social Simulation*, 1-1.

Cardoso, H., & Oliveira, E. (2007). Institutional reality and norms : specifying and monitoring agent or ganizations. *International Journal of Cooperative Information Systems, 16*(1), 67-95.

Cardoso, H., & Oliveira, E. (2009). Adaptive Deterrence Sanctions in a Normative Framework. *Proceedings of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 36–43.

Chebout, M., Mokhati, F., & Badri, M. (2015). Assessing the Effect of Aspect Refactoring on Multi-Agent Applications: A Dynamic Analysis. *International Journal of Agent Technologies and Systems, 7*(3), 45-66. doi:10.4018/IJATS.2015070103

Chebout, M., Mokhati, F., Badri , M., & Babahenini, M. (2019). Monitoring Open Multi-Agent Systems : An Apsect-oriented programming based approach. *Multiagent ann Grid Systems, 15*(2), 1-24.

Chebout, M., Mokhati, F., Badri, M., & Babahenini, M. c. (2016). Towards Preventive Control for Open MAS : An Aspect-based Approach. *In Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 1*, 269-274. doi:10.5220/0006005602690274

Chen, F., & Roşu, G. (2005). Java-MOP: A Monitoring Oriented Programming Environment for Java. *In: Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 546–550.

Chopinaud, C., Seghrouchni, A., & Taillibert, P. (2006). Prevention of Harmful Behaviors Within Cognitive and Autonomous Agents. *17th European Conference on Artificial Intelligence*, 205–209.

Coleman, J. (1998). *Foundations of Social Theory.* Harvard University Press.

Collis, J., Ndumu, D., Nwana, H., & Lee, L. (1998). The ZEUS Agent Building Tool-kit. *BT Technology Journal*, 60 - 68.

Criado , N. (2013). Using norms to control open multi-agent systems. *AI Communications*, 317-318.

Criado, N., Argente, E., Noriega, P., & Botti, V. (2013). MaNEA: A distributed architecture for enforcing norms in open MAS. *Engineering Applications of Artificial Intelligence, 26*(1), 76-95.

da Silva Figueiredo, K., Torres da Silva, V., & de Oliveira Braga, C. (2011). Modeling Norms in Multi-agent Systems with NormML. *In: De Vos M., Fornara N., Pitt J.V., Vouros G. (eds) Coordination, Organizations, Institutions, and Norms in Agent Systems VI. COIN 2010. Lecture Notes in Computer Science* (pp. 39-57). Springer, Berlin, Heidelberg.

Dastani, M., & Torre, L. (2004). Programming BOID-plan agents deliberating about conflicts among defeasible mental attitudes and plans. *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)* (pp. 706–713). New York, NY, USA: IEEE Computer Society.

Dastani, M., Dignum, V., & Dignum, F. (2003). Role Assignment in Open Agent Societies. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS'2003)*, 489-496.

Deguet, J. (2008). Intégration de l'émergence au sein des systèmes multi-agents. Une étude appliquée à la recherche heuristique. *Thèse de doctorat de l'Université Joseph Fourier*.

Delahaye, M. (2007). *Instrumentation de code Java ; Étude bibliographique.* IFSIC : Institut de formation supérieurs en informatique et communication, Université de rennes.

Demazeau, Y. (1995). From interactions to collective behavior in agent-based systems. *Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo*, 117-132.

Dignum, F. (1999). Autonomous agents with norms. *Artificial Intelligence and Law*, 69–79.

Dignum, F., Morley, D., Sonenberg, L., & Cavedon, L. (2000). Towards socially sophisticated BDI agents. *Proceedings Fourth International Conference on MultiAgent Systems.* Boston, USA: IEEE.

Dignum, V., Meyer, J.-j., Weigand, H., & Dignum, F. (2002). An organization-oriented model for agent societies. *Proceedings of International Workshop on Regulated Agent-Based Social Systems: Theories and Applications (RASTA'02), at AAMAS.* Bologna, Italy.

Doan Van Bien, D., Lillis, D., & Collier, R. (2010). Call Graph Profiling for Multi Agent Systems. *In: Dastani M., El Fallah Segrouchni A., Leite J., Torroni P. (eds) Languages, Methodologies, and Development Tools for Multi-Agent Systems. LADS 2009. Lecture Notes in Computer Science, 6039*, 153-167. doi:https://doi.org/10.1007/978-3-642-13338-1_9

Doan Van Bien, D., Lillis, D., & Collier, R. W. (2010). Space-Time Diagram Generation for Profiling Multi Agent Systems. *In: Braubach L., Briot JP., Thangarajah J. (eds) Programming Multi-Agent Systems. ProMAS 2009. Lecture Notes in Computer Science, 5919*, 170-184. doi:https://doi.org/10.1007/978-3-642-14843-9_11

Dufour, B., Goard, C., Hendren, L., de Moor, O., Sittampalam, G., & Verbrugge, C. (2004). Measuring the dynamic behaviour of AspectJ programs. *Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '04)*, 150-169 .

Ernst, M. (2003). Static and Dynamic Analysis: Synergy and Duality. *Proceedings of the 1st ICSE Workshop on Dynamic Analysis*, 24 - 27.

Esteva, M., Rodríguez-Aguilar, J.-A., Sierra, C., Garcia, P., & Arcos, J. (2001). On the Formal Specification of Electronic Institutions. *In: Dignum F., Sierra C. (eds) Agent Mediated Electronic Commerce. Lecture Notes in Computer Science,* , 126-147.

Esteva, M., Rosell, B., Rodriguez-Aguilar , J., & Arcos, J. (2004). AMELI: an agent-based middleware for electronic institutions. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, 236–243.

Etzioni, O., & Weld, D. (1994). A softbot-based interface to the intemet. *Communications of the ACM*, 72-76.

Fagundes, M., Ossowski, S., & Meneguzzi, F. (2014). Analyzing the tradeoff between efficiency and cost of norm enforcement in stochastic environments. *Proceedings of the Twenty-first European Conference on Artificial Intelligence (ECAI'14)*, 1003-1004 .

Felicíssimo, C., Chopinaud, C., Briot, J.-P., Seghrouchni, A., & Lucena, C. (2008). Contextualizing normative open multi-agent systems. *Proceedings of 23rd Annual ACM Symposium on Applied Computing (SAC 2008)*, 52-59.

Felicíssimo, C., Lucena, C., Carvalho, G., & Paes, R. (2005). Normative Ontologies to Define Regulations Over Roles in Open Multi-agent systems. *AAAI Fall Symposium "Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems", Technical Report FS-05-08. USA*.

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

Ferber, J. (2009). MadKit pas à pas. *Support technique de la pletforme MaDKit LIRMM - Université de Montpellier II*.

Ferber, J. (2006). Introduction aux concepts et méthodologies de conception multi-agents. *In : Amblard F. Phan D. eds.(2006) Modélisation et simulation multi-agents pour les Sciences de l'Homme et de la Société : une introduction, Londres, Hermes-Sciences, 414 p. ISBN : 2-7462-1310-9. chapitre 1*, 11-36.

Ferber, J., & Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. *In Proceedings of ICMAS'98, IEEE Computer Society Press*, 128-135.

Ferber, J., Gutknecht, O., & Michel, F. (2004). From Agents to Organizations: An Organizational View of Multi-agent Systems. *In: Giorgini P., Müller J.P., Odell J. (eds) Agent-Oriented Software Engineering IV. AOSE 2003. Lecture Notes in Computer Science, 2935*, 214-230. doi:https://doi.org/10.1007/978-3-540-24620-6_15

Ferber, J., Michel, F., & Baez-Barranco , J.-A. (2004). AGRE: Integrating Environments with Organizations. *In: Weyns D., Van Dyke Parunak H., Michel F. (eds) Environments for Multi-Agent Systems. E4MAS 2004. Lecture Notes in Computer Science, 3374*, 48-56. doi:https://doi.org/10.1007/978-3-540-32259-7_2

Figueiredo, K., & Silva , V. (2011). Norm-ML - A Modeling Language to Model Norms. *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence. (ICAART 2011)*, (pp. 232-237).

Finnemore, M., & Sikkink, K. (1998). International norm dynamics and political change. *International Organization*, 887–917.

FIPA. (2000). Récupéré sur FIPA: http://www.fipa.org/specifications/index.html

Fogués, R., Alberola, J., Such, J., & García-Fornes, A. (2010). Towards dynamic agent interaction support in open multiagent systems. *In: Proceedings of the 2010 conference on artificial intelligence research and development: proceedings of the 13th international conference of the Catalan association for artificial intelligence*, 89–98.

Forgy, C. (1982, September). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence, 19*(1), 17-37.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid :enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, 200-222.

Friedman-Hill, E. (2003). *Jess, The Expert System Shell for the Java Platform.* Livermore, CA: Sandia National Laboratories.

Gaertner, D., Garcia-Camino, A., Noriega, P., Rodriguez-Aguilar, J.-A., & Vasconcelos, W. (2007). Distributed norm management in regulated multiagent systems. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 624–631.

Galán, J., & Izquierdo, L. (2005). Appearances Can Be Deceiving: Lessons Learned Re-Implementing Axelrod's 'Evolutionary Approach to Norms'. *Journal of Artificial Societies and Social Simulation, Journal of Artificial Societies and Social Simulation*, 1-2.

Galliers , J. (1988). A theoretical framework for computer models of cooperative dialogue, acknowledging multiagent conflict. *Thèse de doctorat de l'université Open University, UK*.

García-Camino, A., Noriega, P., & Rodríguez-Aguilar, J. (2005). Implementing Norms in Electronic Institutions. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 667-673.

Gasser , L. (1992). An overview of DAI. *in Gasser, L. and Avouris, N.M. eds. Distributed Artificial Intelligence: Theory and Praxis, Kluwer Academic Publishers*, 9-30.

Gasser, L., Braganza, C., & Herman, N. (1987). Implementing distributed AI systems using MACE. *Proceeding of the Third IEEE Conference on Artificial Intelligence Applications*, 315-320.

Gonzalez-Palacios, J., & Luck, M. (2006). Towards Compliance of Agents in Open Multi-agent Systems. *In: Choren R., Garcia A., Giese H., Leung H., Lucena C., Romanovsky A. (eds) Software Engineering for Multi-Agent Systems V. SELMAS 2006. Lecture Notes in Computer Science, 4408*, 132-147.

Grizard, A., Vercouter, L., Stratulat, T., & Muller, G. (2006). A Peer-to-Peer Normative System to Achieve Social Order. *In Coordination, Organizations, Institutions, and Norms in Agent Systems II : AAMAS 2006 and ECAI 2006 International Workshops, COIN 2006 Hakodate, Japan,May 9, 2006 Riva del Garda, Italy*, 274–289.

Guessoum, Z., & Mandiau, R. (2012). Systèmes multi-agents et Simulation. Dans P. Marquis, J.-M. Ogier, & F. Sèdes, *INFORMATION INTERACTION INTELLIGENCE LE POINT SUR LE I3* (pp. 76 - 120). Cépaduès Editions.

Gutknecht, O. (2001). Proposition d'un modèle organisationnel générique de systèmes multi-agents et examen de ses conséquences formelles, implémentatoires et méthologiques. *Thèse de doctorat de l'université des Sciences et Techniques du Languedoc de Montpellier*.

Gutknecht, O., & Ferber, J. (2000). MadKit: a generic multi-agent platform. *In: Wagner T., Rana O.F. (eds) Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems. AGENTS 2000. Lecture Notes in Computer Science, 1887*. doi:https://doi.org/10.1007/3-540-47772-1_5

Hales, D. (2002). Group reputation supports beneficent norms. *Journal of Artificial Societies and Social Simulation*, 1-4.

Hannoun, M., Sichman, J., & Sayettat, C. (1999). Moise : un modèle organisationnel pour la conception de systèmes multi-agents. *Journées Francophones d'Intelligence Artificielle et Systèmes Multi-Agents (JFIASMA)*, 105-118.

Helsinger, A., Thome, M., & Wright , T. (2004). Cougaar: a scalable, distributed multi-agent architecture. *IEEE International Conference on Systems, Man and Cybernetics* . The Hague, Netherlands: IEEE.

Hewitt, C. (1991). Open information systems semantics for distributed artificial intelligence. *Artificial Intelligence, 47*(1-3), 79-106. doi:10.1016/0004-3702(91)90051-K

Hexmoor, H., Gunnu Venkata, S., & Hayes, D. (2006). Modelling social norms in multiagent systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 19-71.

Hill, E. (2003). *Jess in Action: Java Rule-Based Systems.* United States: Manning Publications

Hollander, C., & Wu, A. (2011). The current state of normative agent-based systems. *Journal of Artificial Societies and Social Simulation, 14*(2), 6.

Hoogendoorn, M., & Treur, J. (2009). An Adaptive Multi-Agent Organization Model Based on Dynamic Role Allocation. *In : International Journal of Knowledge-based and Intelligent Engineering Systems, 13*(3), 119-139. doi: 10.3233/KES-2009-0180

Huynh, T., Jennings, N., & Shadbolt, N. (2006). An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 119-154.

ISO/IEC/IEEE 24765. (2017). *Systems and software engineering – Vocabulary.*

Jennings, N. (2000). On Agent-Based Software Engineering. *Artificial Intelligence, 117*(2), 277-296.

Jennings, N., & Wooldridge, M. (2000). Agent-Oriented Software Engineering. *ARTIFICIAL INTELLIGENCE*, 277--296.

José Plácido da Cunha, F., Ferenzini Martins Sirqueira, T., Leles Viana, M., & José Pereira, C. (2018). Extending BDI Multiagent Systems with Agent Norms. *International Journal of Computer and Information Engineering*, 302-309.

Kálmán, R., Falb, P., & Arbib, M. (1969). *Topics in mathematical system theory.* McGraw-Hill.

Khaled, R., Noble, J., & Biddle, R. (2003). InspectJ: Program Monitoring for Visualisation Using AspectJ. *Proceedings of the 26th Australasian computer science conference (ACSC)*, 359–368.

Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., & G. Grisw, W. (2001). An Overview of AspectJ. *In: Knudsen J.L. (eds) ECOOP 2001 — Object-Oriented Programming. ECOOP 2001. Lecture Notes in Computer Science, 2072*, 327-354. doi:https://doi.org/10.1007/3-540-45337-7_18

Klein, F. (2009). Contrôle d'un sma réactif par modélisation et apprentissage de sa dynamique globale. *Thèse de doctorat de l'université de Nancy 2*.

Knobbout, M. (2016). Logics for Modelling and Verifying Normative Multi-Agent Systems. *Thèse de doctora de l'école : Dutch Research School for Information and Knowledge Systems, Netherlands*.

Laddad, R. (2003). *AspectJ in Action: Practical Aspect-Oriented Programming.* Shelter Island, État de New York, États-Unis: Manning.

Laouadi, M., Mokhati, F., & Seridi-Bouchelaghem, H. (2017). A formal framework for organization-centered multi-agent system specification: A rewriting logic based approach. *Multiagent and Grid Systems, 13*(4), 395–419. doi:10.3233/MGS-170277

Mahmoud, M., Ahmad, M., Yusoff, M., & Mustapha, A. (2014). A Review of Norms and Normative Multiagent Systems. *The Scientific World Journal*, 23.

Mani, N., Garousi, V., & Far, B. (2008). Monitoring Multi-Agent Systems for deadlock detection based on UML models. *In Proceedings of Canadian Conference on Electrical and Computer Engineering*, 1611-1616.

Mansour, S. (2007). Un modèle de gestion distribuée de groupes ouverts et dynamiques d'agents mobiles. *Thèse de doctorat de l'universitée de Pau et des Pays de l'Adour*.

Mansour, S., & Ferber, J. (2007). Agent Groupe Rôle et Service : Un modèle organisationnel pour les systèmes multi-agents ouverts. *Journées Francophones des Systèmes multi-agents, Carcassonne, France*, 107-116.

Marir, T., Silem, A., Mokhati, F., Gherbi, A., & Bali, A. (2019). NorJADE: An Open Source JADE-Based Framework for Programming Normative Multi-Agent Systems. *International Journal of Open Source Software and Processes, 10*(2), 1-20.

Menken, M. (2002, December). *Jess Tutorial*. Récupéré sur https://www.dsi.fceia.unr. edu.ar/downloads/IIA/recursos/jess-tutorial.pdf.

Merriam-webster. (2010). *merriam-webster*. Récupéré sur merriam-webster: https://www. merriam-webster.com/

Meyer, J.-J. (1993). *Deontic logic in computer science: Normative system specification.* Amsterdam, The Netherlands: Roel J. Wieringa.

Meyer, J.-J., & Wieringa, R. (Éds.). (1994). *Deontic logic in computer science: normative system specification.* New York: John Wiley & Sons.

Microsoft, A. (2018). *Microsoft Agent API*. Récupéré sur Microsoft Agent API: https://docs.microsoft.com/en-us/windows/win32/lwef/microsoft-agent-programming-interface-overview

Minar, N., Burkhart, R., Langton, C., & Askenazi, M. (1996). The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations. *Santa Fe Institute Working Pape*, 6 - 42.

Modgil , S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., & Luck , M. (2009). A framework for monitoring agent-based normative systems. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 153-160 .

Morin, E. (1977). *La Nature de la nature.* Le seuil.

Mukherjee, P., Sen, S., & Airiau, S. (2007). Emergence of Norms with Biased Interactions in Heterogeneous Agent. *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops WI-IATW '07*, 512-515.

Nunes, I., Lucena, C., & Luck, M. (2012). Bdi4jade: a bdi layer on top of JADE. *In Proc. of the 9th WS on Programming Multiagent Systems (ProMAS'2011)*, 88–103.

Nusayr , A., & Cook, J. (2009). Using AOP for detailed runtime monitoring instrumentation. *In: Proceedings of the Seventh International Workshop on Dynamic Analysis (WODA)*, 8-14.

Nwana, H., Ndumu, D., Lee, L., & Collis, J. (1999). Zeus: A toolkit for building distributed multi-agent systems. *in Applied Artificial Intelligence, 13*(1-2), 129-185. doi:10.1080/ 088395199117513

Ogata, K. (2009). *Modern Control Engineering (5th Edition).* Prentice-Hall.

Online Encyclopaedia Britannica . (2012). Récupéré sur Online Encyclopaedia Britannica : http://www.britannica.com/EBchecked/topic/418203/norm

Paes, R., Carvalho, G., Lucena, C., Alencar, P., Almeida, H., & Silva, V. (2005). Specifying Laws in Open Multi-Agent Systems. *In: Agents, Norms and Institutions for Regulated Multi-agent Systems (ANIREM)*.

Pawlak, R., Retaillé, J.-P., & Seinturier, L. (2004). *Programmation orientée aspect pour Java/J2EE.* Paris : Eyrolles.

Pearce, D., Webster, M., Berry, R., & Kelly, P. (2007). Profiling with AspectJ. *Software Practice and Experience, 37*(7), 747-777.

Peterson, C. (2011). La logique déontique: Une application de la logique à l'éthique et au discours juridique. *mémoire présenté pour avoir la maitrise universitaire en philosophie de l'université de de Montréal*.

Piskorski, M., & Gorbatai, A. (2017). Testing Coleman's Social-Norm Enforcement Mechanism: Evidence from Wikipedia. *American Journal of Sociology*, 1183-1222.

Plattner, B., & Nievergelt, J. (1981). Monitoring Program Execution: A Survey. *IEEE Computer*, 76-93.

Richters, M., & Gogolla, M. (2003). Aspect-oriented monitoring of UML and OCL constraints. *In: Proceedings of the 4th Workshop on Aspect-Oriented Modeling with UML on the 6th International Conference on the Unified Modeling Language (UML)*.

Rosenschein, J., & Genesereth, M. (1985). Deals Among Rational Agents. *In Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, 91-99.

Rotolo, A., & Van der Torre, L. (2011). Rules, Agents and Norms: Guidelines for Rule-Based Normative Multi-Agent Systems. *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, 53-66.

Sabetta, A., & Koziolek, H. (2008). Measuring Performance Metrics: Techniques and Tools. *Dependability Metrics, Springer*, 226- 232.

Sandia National Laboratories. (2013, November). *Jess,the rule engine for java*. Récupéré sur https://herzberg.ca.sandia.gov/jess/.

Savarimuthu, B. T. (2011). Mechanisms for norm emergence and norm identification in multi-agent societies. *Thèse de doctorat de l'université de Otago, Dunedin, New Zealand*.

Savarimuthu, B., Purvis, M., Purvis, M., & Cranefield, S. (2009). Social Norm Emergence in Virtual Agent Societies. *International Workshop on Declarative Agent Languages and Technologies*, 18-28.

Schwabe, D. (2001). A Conference Review System. *1st Workshop on Web-oriented Software, Valencia*, -. Récupéré sur http://www.dsic.upv.es/~west2001

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 51-92.

Shoham, Y., & Tennenholtz, M. (1995). On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 231-252.

Sichman, J. (1995). Du raisonnement social chez les agents : une approche fondée sur la théorie de la dépendance. *Thèse de doctorat, Institut National Polytechnique de Grenoble*.

Sichman, J., & Conte, R. (1998). On Personal and Role Mental Attitudes: A Preliminary Dependence-Based Analysis. *In: de Oliveira F.M. (eds) Advances in Artificial Intelligence. SBIA 1998. Lecture Notes in Computer Science, vol 1515. Springer, Berlin, Heidelberg*, 1-10.

Silva, V., Braga, C., & Figueiredo, K. (2011). A Modeling Language to Model Norms. *In: De Vos M., Fornara N., Pitt J.V., Vouros G. (eds) Coordination, Organizations, Institutions, and Norms in Agent Systems VI. COIN 2010. Lecture Notes in Computer Science* (pp. 39–57). Berlin, Heidelberg: Springer.

Silva, V., Duran, F., Guedes, J., & Lucena, C. (2007). Governing multi-agent systems. *Journal of the Brazilian Computer Society*, 19-34.

SnifferJade. (s.d.). *http://jade.tilab.com/documentation/tutorials-guides/sniffer/introduction/*. Récupéré sur http://jade.tilab.com/.

Sontag, E. (2013). *Mathematical Control Theory : deterministic finite dimensional systems (2nd ed).* Springer-Verlag New York.

Stratulat, T. (2002). Systèmes d'agents normatifs: concepts et outils logiques. *Thèse de doctorat de l'université de Caen / Basse-Normandie*.

Such, J., García-Fornes, A., Espinosa, A., & Bellver, J. (2013). Magentix2: A privacy-enhancing Agent Platform. *Engineering Applications of Artificial Intelligence*, 96-109.

Valero, S., del Va, E., Alemany, J., & Botti, V. (2015). Using Magentix2 in Smart-Home environments. *In: Herrero Á., Sedano J., Baruque B., Quintián H., Corchado E. (eds) 10th International Conference on Soft Computing Models in Industrial and Environmental Applications. Advances in Intelligent Systems and Computing* (pp. 27-37). Springer, Cham.

Vázquez-Salceda, J., Aldewereld, H., & Dignum, F. (2005). Norms in multiagent systems: From theory to practice. *Computer Systems Science and Engineering*.

Vercouter, L. (2000). Conception et mise en oeuvre de systèmes multi-agents ouverts et distribués. *Thèse de doctorat de l'Université Jean Monnet et de l'Ecole des Mines de Saint-Etienne*.

Vercouter, L. (2001). Une Gestion Distribuée de l'Ouverture dans un Système Multi-Agent. *Journées Francophones d'Intelligence Articifielle Distribuée et des Systèmes Multi-Agent (JFIADSMA'01), Hermes, Montréal, Canada*.

Vercouter, L. (2004). MAST : Un modèle de composants pour la conception de SMA. *Journée Multi-Agents et Composants (JMAC'04), Paris, France*.

Vercouter, L., & Muller, G. (2010). L.I.A.R.: Achieving Social Control in Open and Decentralized Multiagent Systems. *Applied Artificial Intelligence*, 723–768.

Verhagen, H. (2001). Norms and artificial agents. *Proceedings of the 6th Meeting of the Special Interest Group on Agent-Based Social Simulation, ESPRIT Network of Excellence on Agent-Based Computing.* Amsterdam, Holland, November.

Villatoro, D., Sen, S., & Sabater-Mir, J. (2010). Social Norms and Sanctioning: A Game Theoretical Overview. *International Journal of Agent Technologies and Systems (IJATS)* , 1–15.

Viswanathan, D., & Liang, S. (2000). Java virtual machine profiler interface. *IBM Systems Journal*, 82 - 95.

von Scheve, C., Moldt, D., Fix, J., & von Lude, R. (2005). My agents love to conform: emotions, norms, and social control in natural and artificial societies. *in Proceedings of the Symposium on Normative Multi-Agent Systems (NorMAS '05), Hatfield, UK*.

Waller, J. (2014). *Performance Benchmarking of Application Monitoring Frameworks.* Thèse de doctorat, Department of Computer Science, Kiel University.

White, J. (1994). Telescript technology: The foundation for the electronic marketplace. *White paper, General Magic, Inc., 2465 Latham Street, Mountain View, CA 94040*.

Woleński, J. (2016). How deontic logic contributes to the analysis of legal systems : Review of Navarro & RodríGuez, Deontic Logic and Legal Systems (CUP 2014). *Revus - Journal for Constitutional Theory and Philosophy of Law*, 119-122.

Woodside, M., Franks, G., & Petriu, D. (2007). The Future of Software Performance Engineering. *International Conference on Software Engineering, Workshop on the Future of Software Engineering (FOSE 2007). IEEE Computer Society*, 171 – 187.

Wooldridge, M. (2002). Intelligent Agents: The Key Concepts. *Proceedings of the 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001 on Multi-Agent-Systems and Applications II-Selected Revised Papers*, 3-43 .

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems.* WILEY Publisher 2nd Edition.

Wooldridge, M., & Ciancarini, P. (2001). *Agent-oriented software engineering : the state of the art* (Vol. 1957). Berlin, Heidelberg: In: Ciancarini P., Wooldridge M.J. (eds) Agent-Oriented Software Engineering. AOSE 2000. Lecture Notes in Computer Science.Springer.

Wooldridge, M., & Jennings, N. (1995). Agent Theories, Architectures, and Languages : A Survey. *International Workshop on Agent Theories, Architectures, and Languages*, 1–39.

Wooldridge, M., Jennings, N., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 285-312.

Wright, G. (1951). Deontic Logic. *Mind, 60*(237), 1-15.

Wygant, R. (1989). CLIPS — A powerful development and delivery expert system tool. *Computers & Industrial Engineering*, 546-549.

Xu, H., Zhang, X., & Patel, R. (2007). Developing role-based open multi-agent software systems. *International journal of computational intelligence theory and practice (IJCITP), 2*(1), 39-56.

Yoo, M.-J., & Briot, J.-P. (1999). Une approche componentielle pour la modélisation d'agents coopératifs et leur validation. *Rapport technique Laboratoire d'Informatique de Paris 6 (LIP6)*.

Young, H. (1993). The evolution of conventions. *Econometrica, Econometric Society*, 57-84.

Young, H. (2008). *Social norms.* in The New Palgrave Dictionary of Economics, S. N. Durlauf and L. E. Blume, Eds., Palgrave Macmillan, New York, NY, USA.

Zabczyk, J. (1995). *Mathematical control theory : an introduction.* Birkhäuser Basel.