Towards Preventive Control for Open MAS
An Aspect-based Approach

Mohamed Sedik Chebout[1, 2], Farid Mokhati[2], Mourad Badri[3] and Mohamed Chaouki Babahenini[4]
[1]*Department of Mathematics and Computer Science, University of Ouargla, Ouargla, Algeria*
[2]*Department of Mathematics and Computer Science, RELA(CS)2 Laboratory, University of Oum el Bouaghi, Oum el Bouaghi, Algeria*
[3]*Department of Mathematics and Computer Science, Software Engineering Research Laboratory, University of Quebec, Trois-Rivières, Canada*
[4]*Department of Mathematics and Computer Science, LESIA Laboratory, University of Biskra, Biskra, Algeria*

Keywords: Open MAS, Preventive Control, Aspect Paradigm, Profiling.

Abstract: In Open MAS (Open Multi Agent Systems), agents can freely join and leave systems at any time. The inherent specificities of such systems like dynamicity, non-determinism and emergency make their target states difficult to achieve. Agents, in Open MAS, are often heterogeneous, self-interested with conflicting individual goals and limited trust. Consequently, newly entered (external) agents are often considered as a potential disturbance of systems. In this paper, we present a novel preventive control approach based on Aspect-Oriented Programming (AOP) paradigm for mastering Open MAS' behaviour. The proposed control process is mainly accomplished in two steps: (1) Observing agents movements by intercepting all external requests of agents wanting accessing to the system. A request analysis process will be held in terms of compliance capabilities presented by this agent, and (2) Deciding, based on AspectJ constructors, either to allow agents if the capabilities they have enable a possible progress in actual system state to the target state, or prevent it otherwise. The proposed approach is illustrated using a MaDKit-based application.

1 INTRODUCTION

Open multi-agent systems (Open MAS) are societies in which autonomous, heterogeneous and independently designed entities can work towards similar or different ends (Lopez, 2003). An open multi-agent system is a MAS where agents may be added, removed or evolve (e.g. modify their abilities). Moreover, open MAS should support and deal with three possible modifications of their composition (Vercouter, 2000): *Addition* of an agent to the MAS (in order to provide new capabilities), *Removal* of an agent from the MAS (if its capabilities are obsolete) or *Evolution* of an agent (the agent gains new capabilities and loses some of its old capabilities).

Although heterogeneity and autonomy are the required properties of the agents to build open and flexible systems, they are always subject to unanticipated interactions, and increase the vulnerability of the system to the introduction of faulty or malevolent agents, because it is not possible to directly constrain agents' behaviour (Vercouter and Muller, 2010). To avoid that, it is necessary to define and implement control mechanisms to reduce as possible the gap between the observed behaviour and the desired one, i.e.: is it possible for a given system starting from any initial state, to get a given target state in a predefined fixed time? For this, it is essential to use (or to define) control mechanisms to guide the system toward target state.

In addition, controllability plays a crucial role in many control problems such as openness that we are trying to tackle it. Moreover, to ensure this property for a given system, a control process needs, at first time, a monitoring mechanism to measure system behaviour in a given moment (It's about observability). After that, we proceed through means of actions to redirect current measured state for the target one. Accordingly, observability and control are dual aspects of the same problem (Ionete et al., 2006).

In order to reach our goal, we start by identifying the main features relating to open MAS (Table 1). Our aim behind this collection is to focus on a very specific aspect around open MAS. In order to well

Chebout, M., Mokhati, F., Badri, M. and Babahenini, M.
Towards Preventive Control for Open MAS - An Aspect-based Approach.
DOI: 10.5220/0006005602690274
In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2016) - Volume 1*, pages 269-274
ISBN: 978-989-758-198-4

position the current research work, the following table summarizes some characteristics that we consider interesting for our contribution:

Table 1: Some open MAS characteristics in literature.

N°	open MAS Characteristics	ref
1	Agents are owned by various stakeholders with different **aims and objectives**.	Huynh, 2006 and Vercouter, 2000
2	**No** agent **can know everything** about its environment.	
3	The agents are likely to be **unreliable** and **self-interested.**	
4	No **central authority** can control all the agents.	
5	An open system should support the addition or the removal of some **functions after** its design (and generally, **during its execution**).	Vercouter, 2000
6	In open MAS, the **internal aspects of agents are inaccessible**. The **only** accessible information about agents is their **observable behavior** from the exchange of agents' messages.	Paes, 2005a
7	Open multi-agent systems are societies in which autonomous, **heterogeneous** and **independently designed entities** can work towards **similar** or **different** ends.	Silva, 2007

In Open MAS context, MaDKit: Multi-Agent Development KIT (Gutknecht and Ferber, 2000), is a generic multi-agent platform that allows development of open, dynamic and distributed applications based on an organizational model called AGR (Agent, Group, Role) presented in (Mansour and Ferber, 2007). Furthermore, Aspect-Oriented Programming (AOP) was proposed firstly in (kiszales et al., 1997), and since then, it has received considerable attention in many research works. AOP provides a band new modularization technique by encapsulating crosscutting concerns, leading to the production of software systems that are easier to understand, maintain and reuse. Crosscutting concerns are defined as system concerns (such as logging, exception handling, etc.) that crosscut conventional system modules (such as objects, components, and agents) (Kiszales et al, 1997; Kiszales et al., 2001).

The programmer specifies in the AOP language how and where to place the instrumentation (adding new or modifying existing program functionalities and also extending or enhancing the execution of the program to be observable) (Sarrab, 2015). Although its great success as AOP language, AspectJ (Kiczales et al, 2001) is used with an indecent manner for profiling (David et al, 2007).

In this paper, we present a novel preventive control approach based on Aspect-Oriented Programming (AOP) paradigm for mastering open MAS' behaviour. The proposed control process is mainly accomplished in two steps: (1) Observing agents movements by intercepting all external requests of agents wanting accessing to the system. A request analysis process will be held in terms of compliance capabilities presented by this agent, and (2) Deciding, based on AspectJ constructors, either to allow agents if the capabilities they have enable a possible progress in actual system state to the target state, or prevent it otherwise.

The remainder of this paper is organized as follows: In section 2 we give an overview of major related work. In section 3 we present the organizational model and its implementation under MaDKit agent platform. We give an overview of Aspect oriented programming and AspectJ language in section 4. Section 5 is devoted to the presentation of our approach. Finally, conclusion and some future work directions are presented in section 6.

2 STATE OF THE ART

In this section, we present briefly some works targeting the control issue in MAS:

François Klein et al. have proposed in (Klein et al., 2009) an experimental dynamical approach to enhance the control of the global behaviour of a reactive MA using a Markov decision process (MDP). This approach is based on modeling of global system behaviour and the possible transitions between these behaviours that system can adopt, exploited by a control policy defined by machine learning. The control process implemented has the ability to directly change agent behaviour through parameters. However, modeling system behaviour represents a centralized view of system status, which conflict with the distributed nature of MAS.

In the same context, Howard Carolina Felicissimo et al. have proposed in (Felicissimo, 2005a; Felicissimo, 2005b; Felicissimo, 2008) an extended study on automatic regulation of open MAS based on normative generic ontologies:

In (Felicíssimo, 2005a), the authors have proposed a normative ontology-based approach to define regulations over roles in open MAS. The ontology is independent of domain and has five related basic concepts: Role, Norm, Penalty, Action and Place. This ontology's structure provides a

semantic support for agents to base their behaviour according to norms and to reason about action selection.

In (Felicíssimo, 2008) an approach called DynaCROM (DYNAmic Contextual Regulation in formation provision in open MAS) has been proposed for smoothly applying and managing regulations in open MASs as well as for enforcing precise contextual norms. The DynaCROM's main contributions are: "(i) a definition of a top-down classification for contextual norms, which facilitates the tasks of elucidation, organization and management of norm information; (ii) a contextual normative ontology to explicitly represent the semantic of classified norms in a meaningful way (i.e., with a common understanding) for heterogeneous agents; (iii) a definition of a norm composition process, based on ontology-driven rules, that makes it easy to update the system regulation by both evolving norms in a unique resource (an ontology) and by activating particular rules for acquiring customized compositions of contextual norms; and (iv) a solution for enforcing contextual norms". DynaCROM is still a work in progress.

Although these works have considerably forwarded the domain by proposing novel approaches for Open MAS control, they did not deal with preventive control, which allows to avoid as soon as possible falling into an unpredictable situation caused by external agent's heterogeneity and their suspect behaviour.

The approach we propose in this paper focuses, particularly, on the control in open MAS, in order to provide control mechanisms to them. For this reason, our goal is to predict disturbance source caused by the entry of new external agents in the system. These latters share some properties such as: heterogeneity, different aims and objectives, internal architecture that is inaccessible, etc. Therefore, we aim to benefit from Aspect-Oriented Programming advantages to avoid any kind of system disturbance caused by external agent's movement.

3 MADKIT AGENT PLATFORM

MaDKit: Multi-Agent Development KIT (Gutknecht and Ferber, 2000), is a generic platform for design and implementation of multi-agent systems. This platform is based on an organizational model (Figure 1), named AGR (Agent, Group, Role) (Ferber et al., 2003), rather than an agent architecture or specific interaction model. MaDKit, also, uses an execution engine wherein each agent is built starting from a microkernel.

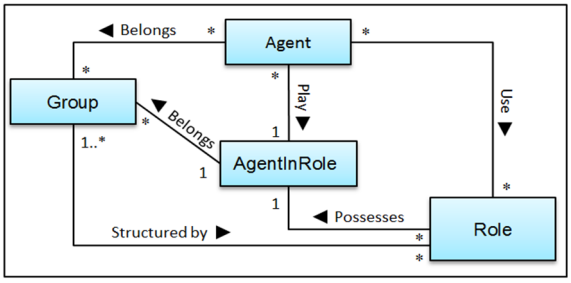

Figure 1: Simplified UML model for AGR (Mansour and Ferber, 2007).

Considering the role definition that represents the primordial concept in the organizational model, (Mansour and Ferber, 2007) have defined role as a sixfold : <Gr, Nm, Cd, Sv, At, Rl> where:

- **Gr** : the group identifier in which the role belongs,
- **Nm** : Role name,
- **Cd :** Role conditions, there are two types :
  - Cdj: conditions that must meet an agent wishing play a role.
  - CdU: conditions that must meet an agent playing a role.
- **Sv** : a set of services defined in the role,
- **At** : Internal attributes of the role,
- **Rl**: a set of rules that govern role.

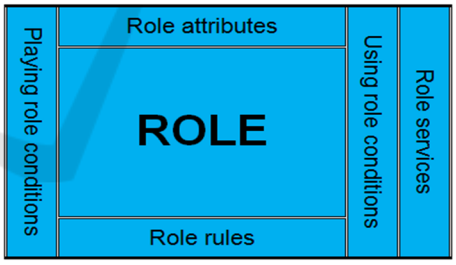

Figure 2: Role's Architecture (Mansour and Ferber, 2007).

We use in the rest of this paper, the term capability rather than service. According to (Mansour and Ferber, 2007), a service is a set of functionality that agent should suggest or ask even basic functions: send/receive message, creation of groupe/role, exploration of acquaintance agent list, etc. Moreover, there are two aspects of reasoning about the capabilities of an agent to achieve its goals, according to (He and Ioerger, 2003) :

- Logical reasoning, which is based on whether the agent can determine a plan that can be used to achieve a goal (i.e. "know-how").

- Quantitative reasoning about levels of skill and whether a set of tasks can be achieved within a specified time and with quality constraints. Capabilities in this sense are determined by the limits of internal processing capacity"

Therefore, capability is an intrinsic functionality attribute. Furthermore, Agent cannot realize either functionality or achieves its goals if it does not have the necessary capabilities. As well, capability is an attribute related much more to performance or how much time and effort required for doing such task, against, service is a concept purely related to design.

4 ASPECTJ FOR PROFILING

AspectJ (Kiczales et al., 2001) is a practical aspect-oriented extension to the Java programming language. AspectJ is the most widely used aspect-oriented programming language; it supports the definition of aspects, advices, join points, and pointcuts. Join points (Kiczales, 2001) are well-defined points in the execution of the program; pointcuts are collections of join points; advices are special method-like constructs that can be attached to pointcuts and aspects which are modular units of crosscutting implementation, comprising pointcuts, advice, and ordinary Java member declarations.

For more information and details about AspectJ constructs, the reader is invited to consult (Laddad, 2009), (Apel and Batory, 2010).

Our proposal for observing open MAS behaviour is to use AspectJ for profiling MAS specific attributes (i.e. dynamic analysis). This choice is motivated by the fact that traditional profilers like (JVMPI: *Java Virtual Machine Profiler Interface*) (Liang and Viswanathan, 1999) and its successor (JVMTI: *Java Virtual Machine Tool Interface*) (O'Hair, 2006) present some drawbacks: (1) it is a fixed interface and, as such, can only enable predefined types of profiling; (2) enabling the JVMPI often dramatically reduces performance. Thus, in AspectJ, the programmer specifies how and where to place the instrumentation. AspectJ also allows for both compile time and load time instrumentation and makes use of BCI (ByteCode Instrumentation) libraries to instrument the application.

In (David et al., 2007), the authors have demonstrated that AspectJ, after investigating four common profiling problems (heap usage, object life time, wasted time and time-spent), was sufficiently flexible to support the four profiling examples. In addition, it was reasonably efficient in most cases, of course with some uncovered limitations according to (David et al., 2007), like: load-time weaving standard libraries, state association, array allocation join point and synchronization.

5 PROPOSED APPROACH

Our proposal (Figure 3) is to control a priori open MAS by intercepting all external requests from agents that want to enter in the system. As it is quoted above, open MAS share properties according to agent's heterogeneity, different agent's aims and objectives. This explains that the integration of a new agent in the system might be a potential source of current state disruption, and therefore emergent behaviour will take place.

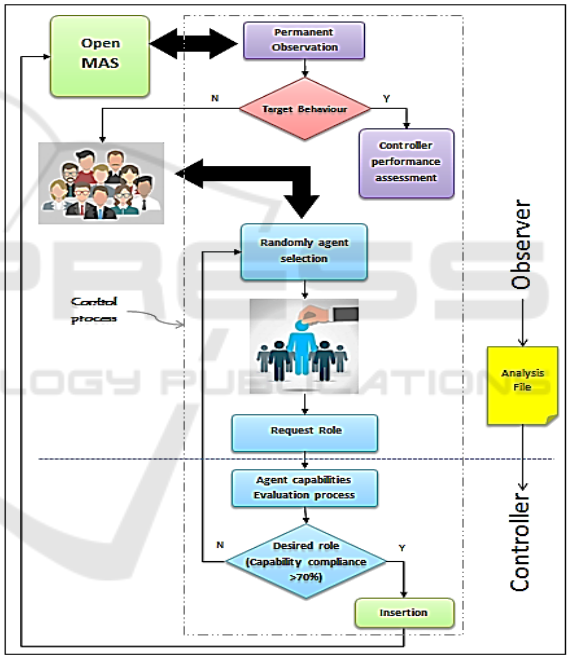

Figure 3: The methodologies of our approach.

In order to avoid falling into such situation, we propose to implement an *observer aspect*, which is responsible for the permanent monitoring of the entry of new agents, once this last wants to get in system.

On the one hand, observer aspect represents, actually, a part of control process, which is also composed of a module that deals with evaluation of agent capabilities; this last, is the core of the control process. On the other hand, agent's role request is composed, as it is mentioned in section 3, by a set of fields. We focus in our context on Cd (Role Condition) especially CdU field that denote

conditions that must meet an agent playing a role. In this case, Agent capabilities evaluation process analyses the compliance of role in terms of agent capabilities. We suppose that provided role by agent includes several capabilities, every capability is associated with a control degree. We assume, also, that we cannot judge the reliability of the capability offered by the agent (Table1, characteristic n°3).

After that, if the rate of agent capability reaches certain level (we choose initially a considerable rate: up then 70%), the agent is allowed to enter, otherwise, it will be prevented using a specific aspect. As well, capabilities evaluation process must analyse another agent's request and so on.

The control process proceeds until the target system state is reached. According to (Klein et al., 2009), current system state measurement, depend strongly of the studied system. Therefore, each system has a different configuration compared to other systems. Thus, we adopt a performance measure related to control time, i.e.: the control process elapsed time to master the behaviour of the studied system.

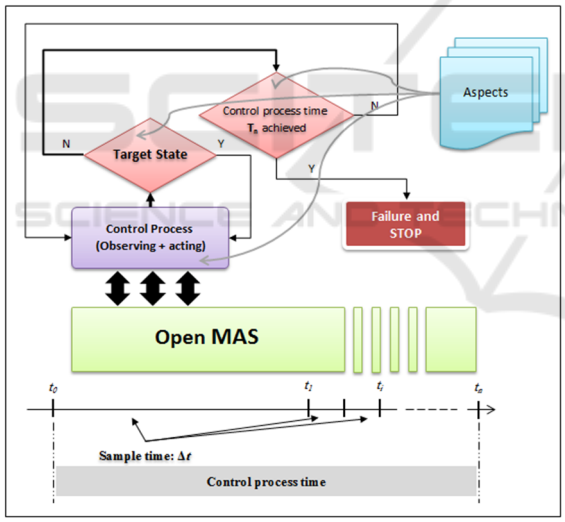

Figure 4: Control process time.

For that, we propose to implement an aspect named *profiler Aspect*. This latter enables intercepting the first call to the first explicit agent requesting role. Then, we will capture current state each sample time (Figure 4), to compare it with the target state (i.e expected system result). We check also, if the total control time is reached without getting desired result, the preventive control process present a failure, and in this case the system no longer has the controllability property.

6 CONCLUSION AND FUTURE WORK

In this paper, we have studied the controllability of Open multi-agent system, using Aspect oriented programming paradigm under MaDKit platform.

Our approach consists, initially, to predict any potential source of system disturbance caused by openness property, by analyzing system dynamics in terms of external agents that get in the system, through profiling mechanism using AspectJ language. Thus, we exhibit that AspectJ has demonstrated its efficiency in such situation against existing profilers that limit profiling types in a very reduce number and its negative impact on analysed system performance. Furthermore, the controller uses the power of AspectJ constructs to act in a way to enable or block all requests coming from entering agents, having a certain degree of impeding of current system state.

In fact, this work is in progress, and the approach explained in this paper represents a first step of a novel open MAS control process. As future work, we plan to extend the proposed approach in order to support more features relating to open MAS control, especially, posteriori control, using reinforcement learning, or simply, how to endow agents with learning mechanisms for achieving their goals?

REFERENCES

Apel, S., Batory, D. 2010. 'How aspectj is used: an analysis of eleven aspectj programs'. Journal of Object Technology (JOT), 117-142.

David, J, P., Matthew, W., Robert, B ., Paul, H, J, K., 2007 . 'Profiling with AspectJ', *Software—Practice & Experience*, 37(7), 747-777.

Felicíssimo, C., Lucena, D., Carvalho, G., Paes, R., 2005. 'Normative ontologies to define regulations over roles in open multi-agent systems' , *In Proceedings AAAI Fall Symp, 171 -176*.

Felicíssimo, C., Lucena, D., 2005. 'An approach to regulate open multi-agent systems based on a generic normative ontology' In *Proceedings of the 1st Workshop on Software Engineering for Agent-oriented Systems (SEAS)*.

Felicíssimo, C., Chopinaud, C., Briot, J, P., Seghrouchni, A., Lucena, C., 2008. 'Contextualizing normative open multi-agent systems'. In *Proceedings of 23rd Annual ACM Symposium on Applied Computing (SAC 2008)*, 1, 52-59.

Ferber, J., Gutknecht, O., Michel, F., 2004. 'From Agents to Organizations, an Organizational View of Multi-Agent Systems'. *Agent-Oriented Software Engineering*

(AOSE) IV, *P. Giorgini, Jörg Müller, James Odell, eds, Melbourne*, July 2003, LNCS 2935, 214-230.

Gradecki, J, D., Lesiecki, N., 2003. 'Mastering AspectJ: Aspect- Oriented Programming in Java'. Wiley Publishing, Inc., Indianapolis, Indiana.

Gutknecht, O., Ferber, J., 2000. 'MaDKit : une architecture de plate-forme multi-agent générique'. *Rapport de recherche, Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRM)*, Université de Montpellier II.

He, L., and Ioerger, T, R., 2003. 'A quantitative model of capabilities in multi-agent systems'. In *Proceedings of International Conference on Artificial Intelligence*, 730-736.

Huynh, T. D., Jennings, N. R., Shadbolt, N. R. 2004b. 'FIRE: An integrated trust and reputation model for open multi-agent systems'. In *Proceedings of the 16th European conference on artificial intelligence (ECAI)*, 18–22.

Ionete, C., Cela, A., Gaid, M, B., 2006. 'Controllability and observability of input/output delayed discrete systems', In *Proceedings of IEEE Amer. Control Confeerence*, 3513-3518.

Kiczales, G.,Hilsdale, E., Hugunin, J., Kersten, M., Palm, J. & Griswold, W.G., 2001. 'An Overview of aspectJ'. In proceedings *of the European Conference on Object-Oriented Programming*.

Kiczales, G., Lamping, J., Menhdekar, A., Maeda, C., Lopes, C., Loingtier, J-M., Irwin, J., 1997. 'Aspect-oriented programming'. In *Mehmet Aksit and Satoshi Matsuoka, editors, European Conference on Object-oriented Programming*, Vol 1241 of Lecture Notes in Computer Science, 220–242. Springer.

Klein, F., Bourjot, C, Chevrier, V., 2009. 'Contribution to the Control of a MAS's Global Behaviour: Reinforcement Learning Tools'. *Alexander Artikis and Gauthier Picard and Laurent Vercouter. Engineering Societies in the Agents World IX - 9th International Workshop, ESAW 2008, Saint-Etienne, France, September 24-26, 2008, Revised Selected Papers, 5485, Springer-Verlag Berlin Heidelberg*, P.173-190, Lecture Notes in Computer Science, 978-3-642-02561-7. <10.1007/978-3-642-02562-410>. <inria-00400348>

Laddad, R., 2009. 'AspectJ in Action' . Enterprise AOP with Spring Applications. Manning, 2nd edition.

Liang, S., Viswanathan, D., 1999. 'Comprehensive profiling support in the Java Virtual Machine'. *Proceedings of the USENIX Conference On Object Oriented Technologies and Systems*. USENIX Association: Berkeley, CA, 229–240.

López, F., 2003 'Social Power and Norms : Impact on agent behaviour'. *PhD thesis*, Univ. of Southampton.

Mansour, S., Ferber, J., 2007. 'Agent Groupe Role, et Service: Un modèle organisationnel pour les systèmes multi-agents ouverts'. *JFSMA07*.

O'Hair, K., 2006. 'The JVMPI transition to JVMTI'. *http://java. sun.com/developer /technicalArticles/ Programming/jv mpitransition/*.

Paes, R., Carvalho, G.R., Lucena, C.J.P., Alencar, P.S.C., Almeida, H.O., Silva, V.T. 2005a. 'Specifying Laws in Open Multi-Agent Systems'. In *Agents, Norms and Institutions for Regulated Multi-agent Systems* (ANIREM), AAMAS.

Sarrab, M. 2015. 'Bytecode instrumentation mechanism for monitoring mobile application information flow'. In *International Journal on Security and Networks, 10(3).*

Silva, V., Duran, F., Guedes, J., & Lucena, C. 2007. 'Governing multi-agent systems'. Journal of Brazilian Computer Society. *Special Issue in Software engineering for multi-agent systems, 13*(2).

Vercouter L., 2000. 'A Distributed Approach to Design Open Multi-agent Systems'. In 2nd *International* Workshop Engineering Societies in the Agents' World (ESAW).

Vercouter, L., Muller, G., 2010. 'L.I.A.R.: Achieving social control in open and decentralized multiagent systems'. Applied Artificial Intelligence: An International Journal, 24(8), 723-768, DOI: 10.1080/ 08839514.2010.499502.

Wang, Q., Gao, H., Alsaadi, F and Hayat, T. 2014. 'An overview of consensus problems in constrained multi-agent coordination'. Systems Science and Control Engineering, 2(1), 275.