Agent Based Modeling of Complex System with AML and the Situation Calculus

Thibaud Brocard Fabrice Bouquet Alain Giorgetti Christophe Lang

1 Context

The aim of this paper is to provide a method to create an agent-based model for a complex system and a way to check some of its properties. This work takes place in a research project[1] where complex systems are next generation microfactories. This project brings together different scientific domains such as computer science, physics, electronics and automatics. The rationale of the project is to define models and tools that will support the collaboration of these different research domains in a common application case. In this paper we focus on the system complexity and put aside the *micro* aspect.

We consider a system as complex when we can not predict its whole behaviour (e.g. due to the high number of entities composing it, their cognition, ...). The system taken as an example is a robot production factory. Robot parts are built in different production cells, then finally assemblied together. Each cell is composed of several tools, actuators, effectors (such as a painter, vernisher or solderer arm) or even other simpler cells. Furthermore the cell number, composition and organization can change dynamically and in an autonomous way. The system is self-organized and adapts its own structure and behaviour in order to fulfill assigned tasks. Instead of a clever, omniscient and single entity managing the factory, we follow the *divide and conquer* paradigm by considering each physical system entity as a rational entity. The Multi-Agent System (MAS) paradigm is, obviously, very close to this approach of modeling complex systems.

We have not yet found a single language and formalism allowing us to describe every aspect of such systems. For this reason our model is divided in three different views: structure, action and interaction. The first two views have their own language and interactions are expressed with a combination of these two languages. We will present each view separately and conclude with implementation results and perspectives.

2 Structure model: AML

The structural part of the model describes the different agents composing the system and the hierarchical way they are organized. Following several works showing that the agents organization in a MAS has a significant impact on the system performances, we use the AGR (for Agent, Group

[1]Project supported by the Franche-Comté region council, see `http://lifc.univ-fcomte.fr/~philippe/site/projets/musine/musine.html` (in french) for details.

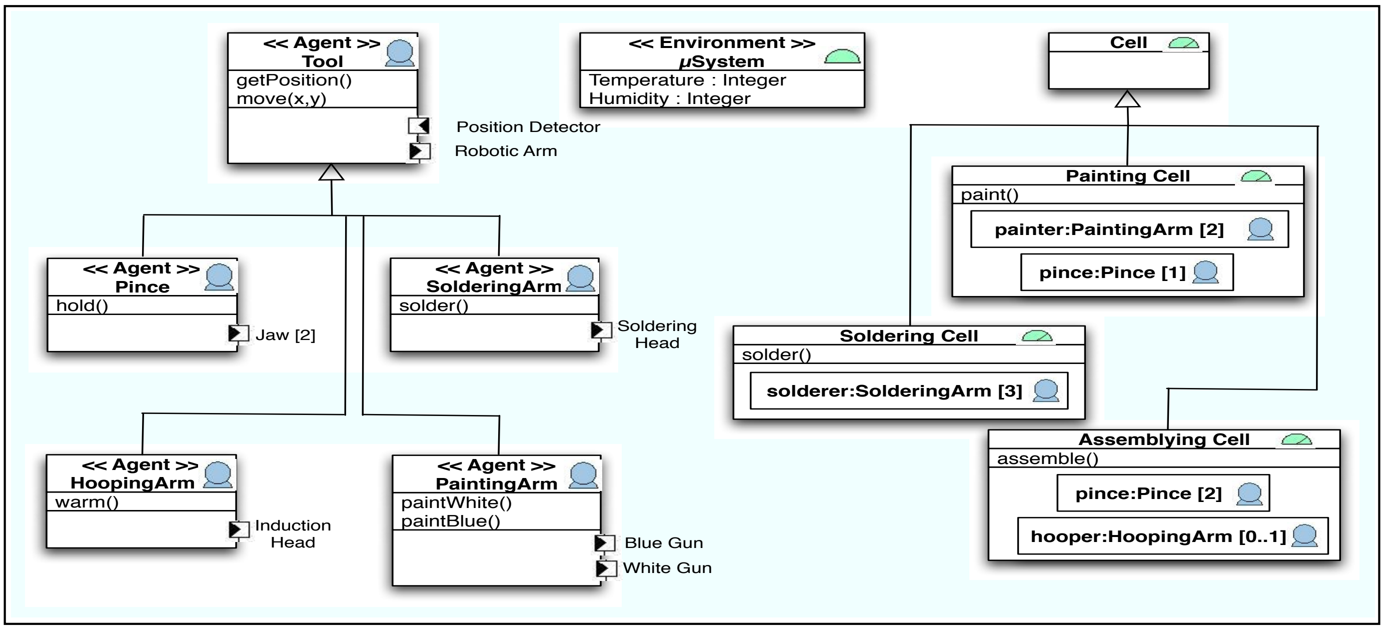

Figure 1: AML structure model

and Role) model presented by Ferber [3] and organize the agents composing the complex system in an holarchy [7, 4], thus putting organization at the foreground.

Most of the time, UML is used to describe MAS, but UML models can not easily deal with the different levels we have in a complex system. For this reason, we have been interested in a more recent language: Agent Modeling Language (AML). AML [10, 1] is a semi-formal visual modeling language for specifying, modeling and documenting systems that incorporate features drawn from MAS theory. AML is suitable for models with several autonomous, concurrent and/or asynchronous entities. Those entities are able to observe and interact with their environment, using complex interactions and services aggregation. AML can also describe social structures and mental characteristics. With the help of AML we will be able to model agent groups and roles in a very natural way. AML is suitable for Ferber's AGR model. Furthermore (and that is the main reason why we chose it), AML deals with holons, so we will be able to describe an holonic structure in our model.

Figure 1 shows the first two levels of the robot factory structure model, in AML. Each tool has a position and is able to move, as shown in the left part of the figure. Differences between tools come from their abilities: a painting tool has two paint guns and is able to paint, a pince is able to hold, etc. The right part of the figure presents the composition of factory cells. There are three different kinds of cells: the painting ones, the soldering ones and the assemblying ones, differing by their tool composition.

3 Action model: Situation Calculus

An agent logic should allow representing dynamic aspects of agents as individuals, but also of parts of a global system, and reasoning about them. We describe agent actions with the help of the Situation Calculus [9]. It is widely used in Artificial Intelligence for representing a dynamical universe and reasoning about it. This universe is characterized by a conjunction of logical formulas where *actions* perform stage changes observed by *fluents*. A *situation* is not a state, but a finite sequence of actions, starting from an initial situation denoted S_0. In the robot factory, for example the situation:

$$do(cell1, assembly, do(agentA, paintBlue, do(agentB, solder, S_0)))$$

represents the action sequence. From the initial situation S_0, $agentB$ has soldered, then $agentA$ has painted in blue and finally the cell $cell1$ has assembled.

The Situation Calculus, slightly extended, allows us to express the action capacity of agents (depending on their ability, their state and the state of the environment as in [2]), the duration of an action (in order to verify that an agent is not performing two different actions at the same time), their organization (through the group they belong to) and their position in the world.

4 Interaction model

The third part of our model describes the different interactions among agents, either with other agents or with the environment. For the present, the modeled interactions are communications (with a black board or by message sending), reactions to the actions, cooperations of several agents to deal with a *complex* action, scheduling, and knowledge learning. We do not need a new langage to describe those interactions: we can express them with the help of AML and the Situation Calculus.

5 Implementation, results and future work

We made an implementation of the Situation Calculus axioms in Prolog as in [8]. The lists of agents, groups, roles and capabilities of agents are extracted from the AML model. The initial situation comes from an instance diagram defined in AML. The Prolog implementation allows us to check if an action or an action sequence is possible. We can check if an agent is capable to perform an action (if it knows how to perform it, if it is able to perform it and if it is not busy) and then get the beginning and the end of every action in the sequence. This provides us a naïve scheduling. It can also check if a scheduling is possible or not. Our implementation of the Situation Calculus supports the notion of different abstraction levels by considering each action differently according to the level at which it occurs.

Even if there are other properties for the agents and the MASs (e.g. as presented in [5]), we describe in these three views all the *basic* properties, which are usual in MASs. One of the next steps is to express new agent properties, such as the Belief/Desire/Intention model. Another direction is to exhibit general rules for deriving the Prolog implementation from the AML and Situation Calculus models. With such rules, implementation would be automated instead of being done manually. Next step is to provide a MAS simulator from our model based on MadKit [6]. The MadKit software architecture has been built on the AGR model.

References

[1] R. Cervenka and I. Trencansky. *The Agent Modeling Language - AML: A Comprehensive Approach to Modeling Multi-Agent Systems.* Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser Basel.

[2] L. Cholvy, C. Garion, and C. Saurel. Ability in a Multi-agent Context: A Model in the Situation Calculus. In *CLIMA VI*, pages 23–36. Springer, 2005.

[3] J. Ferber, O. Gutknecht, and F. Michel. From Agents to Organizations: an Organizational View of Multi-Agent Systems. In *Agent-Oriented Software Engineering IV 4th International Workshop*, pages 214–230, Melbourne, Australia, July 15 2003.

[4] A. Giret and V. Botti. Holons and agents. *Journal of intelligent manufacturing*, 15:645–659, 2004.

[5] R. Goodwin. Formalizing Properties of Agents. Technical report, 1993.

[6] Olivier Gutknecht and Jacques Ferber. Madkit: a generic multi-agent platform. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*, pages 78–79, New York, NY, USA, 2000. ACM.

[7] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316, December 2004.

[8] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.

[9] R. C. Moore. A Formal Theory of Knowledge and Action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, Norwood, NJ, 1985.

[10] I. Trencansky and R. Cervenka. Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS. *Informatica (Slovenia)*, 29(4):391–400, 2005.