Introducing The Concept of Hyperactivity in Multi Agent Systems

Vineet Nagrath,
Laboratoire Le2i, UMR CNRS 5158,
IUT, Le Creusot, FRANCE
vineet.nagrath@gmail.com

Fabrice Meriaudeau,
Laboratoire Le2i, UMR CNRS 5158,
IUT, Le Creusot, FRANCE
fabrice.meriaudeau@u-bourgogne.fr

Aamir Saeed Malik
University Technology Petronas,
Perak, MALAYSIA
aamir_saeed@petronas.com.my

Olivier Morel
Laboratoire Le2i, UMR CNRS 5158,
IUT, Le Creusot, FRANCE
Olivier.Morel@u-bourgogne.fr

Abstract— **Software Agents are no longer the simple communication gateways for devices to interconnect using one or more networks. With Multi Agent Systems contributing in a wide spectrum of intelligent systems, the Agents are in a more proactive role than just being responsible for passing messages between their respective base systems. Agent Relation Charts and the Hyperactive Transaction Model in general is one of the recent attempts of developing a multi-view design model for Multi Agent Systems. The model has made a clear distinction in the regular and intelligent activities of an agent. Based on these differences, the agents are classified into three main categories named as Passive, Active and Hyperactive Agents. In this paper we first attempt to clearly explain the basis on which distinctions are made in the activities of an agent, and why such a distinction improves the overall design process for the multi agent systems. We then define and demonstrate the three kinds of agents based on the distinctions made in their activities and thus introducing the concept of Hyperactivity in a multi agent system.**

Keywords- Multi Agent Systems, Model Based Engineering, Multi-View Modeling, , Agent Relation Charts, Hyperactive Agents, Software Agents.

I. Introduction

A. Multi Agent Systems

There is no one definition available for Agents which capture the concept of agent in a technically precise manner. Some of the most cited definitions are given by [1], [2] and [3] where agents are defined as autonomous computer systems which work flexibly in a multi-agent system (dynamic, unpredictable and open environments). Agents have individual goals which may or may not be same as the other agents present in the environment and systems where more than one agent is playing are known as multi agent systems. To summarize [4] Agents are autonomous problem solving entities embedded in a particular observable environment with specific role and particular objectives.

Distributed Artificial Intelligence (DAI) has two sub divisions [5] namely distributed problem solving (DPS) and multi agent systems (MAS). Unlike DPS which deals with the distribution of the process of problem solving, MAS highlights the behavioral and interaction related complexities [6]. As multiple agent systems are studied along a wide spectrum of domains, there are several definitions available. We can summarize the widely accepted definitions as follows.

MAS are systems with variable number of agents with particular goals or set of tasks. They interact with each other by flexible and complex protocols. The combined effect of simple competitions and giving equal importance to individual and collective tasks give rise to "the intelligence" in MAS [7]. MAS are thus a network of entities capable of working together to solve problems beyond the capability of any one entity.

B. Software Engineering for Agents

Agent-Oriented Software Engineering (AOSE) proposes to think in an "agent-oriented mindset" to split the problem into agents while a more refined version of AOSE, Agent-Based Software Engineering (ABSE) is more practical towards agent building. But once we have decided to think in terms of agents and have subdivided the problem into agents, the task still remains to write a good design for the agent. A design that is complete as well as close to popular industrial software engineering practices. The design model must also present opportunities for all stake holders to contribute in the design process. MAS can be very complicated and thus require multi-layer abstraction in the design process.

With the emergence of affordable networked computing technologies and the variety of communication media available to them, the new age designers need exact software engineering tools to convert their ideas to designs.

C. Model Based Engineering

The complexity of the robotic/agent systems is increasing. Consumer Demand and business logic pushes the designers to develop flexible, adaptable and good quality products in less time. The software engineering methods for optimizing production directly applies to the robotic/agent products[8] and thus its natural to keep an eye on the software world to gain benefits for the robotic/agent industry.

In this light, in the past 10 years, agent/robotic software have got its influence from the Component based and object oriented software development models [8]. In the same way, the newly appeared model driven engineering (MDE) paradigm [9] has caught the attention of the robotic/agent system developers.

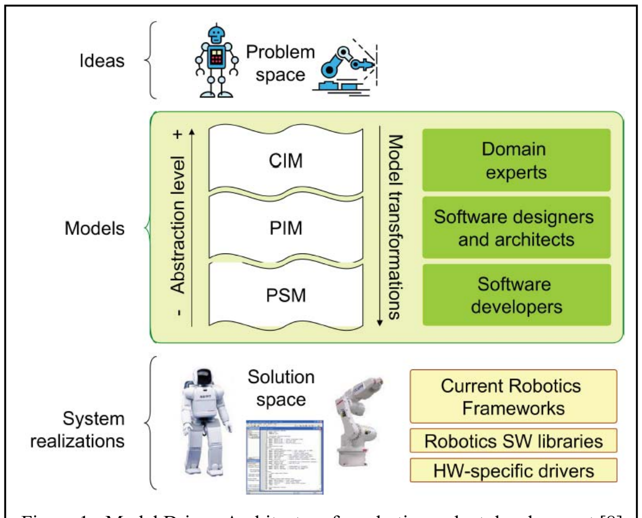

Figure 1. Model Driven Architecture for robotic product development [8].

MDE is a concept driven approach towards design with less focus on the implementational details. The real systems are represented in the design with a very abstract and simple version called its model [10]. In the design process, a number of non-technical players also have to make their contribution. Models, which are a very simple representation of the system goes beyond the technical and implementational knowledge of the system and enables everyone to contribute in the design process. This approach was further strengthened when in the year 2001 the Object Management Group (OMG) published their work on the Model Driven Architecture (MDA) [9]. OMG's MDA had three layered structure "Fig. 1" where Computational independent models (CIM) with maximum abstraction were made for domain experts, Platform-independent Models (PIM) with middle level abstraction were suited for software and system designers and lowest level Platform-Specific Models (PSM) were for implementation level engineers. When the platform is not clearly defined, MDA allows a thin border between PIMs and PSMs.

D. *Multi-View Modeling*

The core concept behind multiple view modeling is representing the design as a set of different views or representations, each one highlighting a particular kind of design element. The advantage of a multi-view model is that it allows simplifying a complex system design and provides various layers of abstraction to various viewers.

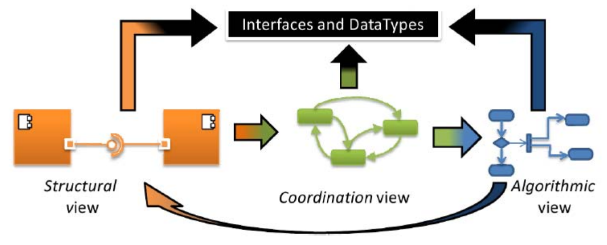

Figure 2. The three views of V3CMM [8].

One example multi view modeling methodology for robotic products is V3CMM, which is a 3-View Component Meta-Model for Model-Driven Robotic Software Development [8]. In this model "Fig. 2" the three views represent different kind of information about the design. The structural view defines the overall structure of the system, the coordination view defines the event-driven functioning and the algorithmic view describes each component's individual working algorithm [8].

II. Hyperactive Transaction Model & Agent Relation Charts (ARCs)

Hyperactive transaction model [11] is an attempt to provide a model for designing multi agent systems with a different "thought process". A similar example can be seen in object oriented software modeling. Object oriented programming is not only a different way to write code, but a new "thought process" to design software where we think in terms of objects in real life before making their equivalent class in the code. Likewise in Hyperactive Transaction Model, the attempt is to think of agents in terms of humans working together in a team.

Like humans have relationships, in HTM, agents have a relation with other agents. There relations have parameters that govern the interaction and trade between the related agents. In HTM, the trade between agents is modeled in a separate view. Once the relations, trade logics and hyperactivity (Discussed in Section III) controls have been designed, the HTM has a behavioral component to model system's response in different use cases and event sequences. HTM is thus composed of five different views "Fig. 3". These views are named as relational, structural, trade, hyperactivity and behavioral view and contain design elements corresponding to their name. In the next section (Section III) we will further discuss these views, specially the hyperactivity view of the model.

ARCs is the computation independent top layer of HTM [11] "Fig. 3". The five views in the hyperactive transaction model gives five different kinds of ARCs. The relational and structural views are combined together as ARC; trade view has a Trade-ARC (TARC); hyperactive view has a Hyperactivity-ARC (HARC) and the behavioral view has two kinds of ARCs, Use Case-ARC (UARC) and Sequence-ARC (SARC).

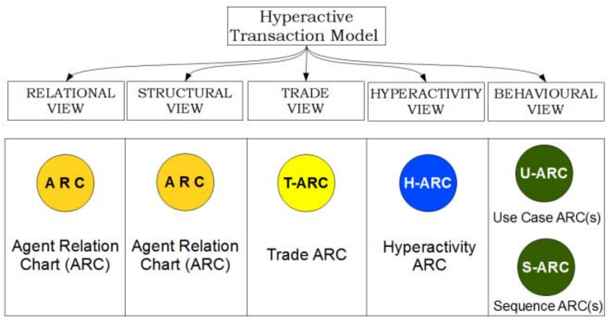

Figure 3. The five Views of HTM and its topmost ARC layer [11].

III. HYPERACTIVITY

In this section we will discuss hyperactivity and all associated terms that we frequently use in the hyperactive transaction model, and Agent Relation Charts [11]. Software agents are by definition software components representing the base software in the network clouds. Agents are a strongly connected component of the base software or separate software running on the host machine. The base software is connected to associated agents (other agents in the network which are associated with the base software for some common goal or service) through one or more kinds of networks (hence the term network clouds is used to cover the possibility of having more than one kind of networks in the Multi Agent System.)

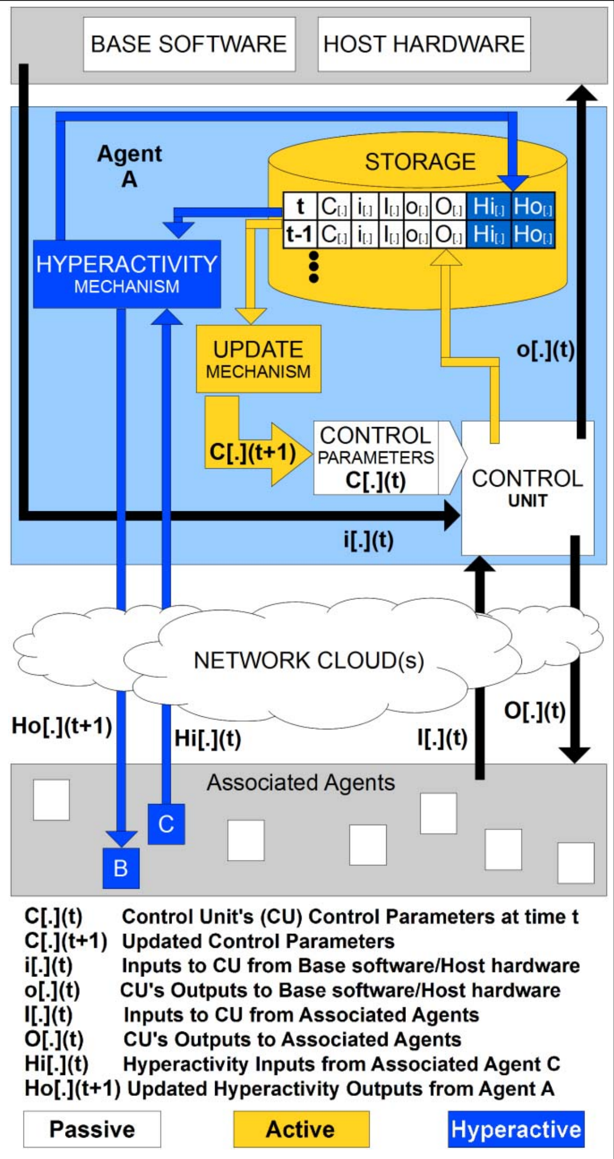
BASE SOFTWARE
HOST HARDWARE
Agent A
STORAGE
t C[.] i[.] I[.] o[.] O[.] Hi[.] Ho[.]
t-1 C[.] i[.] I[.] o[.] O[.] Hi[.] Ho[.]
HYPERACTIVITY MECHANISM
UPDATE MECHANISM
o[.](t)
C[.](t+1)
CONTROL PARAMETERS C[.](t)
CONTROL UNIT
i[.](t)
NETWORK CLOUD(s)
Ho[.](t+1)
Hi[.](t)
I[.](t)
O[.](t)
Associated Agents
B
C
C[.](t) Control Unit's (CU) Control Parameters at time t
C[.](t+1) Updated Control Parameters
i[.](t) Inputs to CU from Base software/Host hardware
o[.](t) CU's Outputs to Base software/Host hardware
I[.](t) Inputs to CU from Associated Agents
O[.](t) CU's Outputs to Associated Agents
Hi[.](t) Hyperactivity Inputs from Associated Agent C
Ho[.](t+1) Updated Hyperactivity Outputs from Agent A
Passive
Active
Hyperactive

Figure 4. Hyperactivity in a Multi Agent System.

A. *Passive Agents*

Looking at the general structure of a software agent in "Fig. 4" we can see the agent as having a control unit in between the lines connecting the base software (and/or host hardware) to other agents via the network clouds (one or more different kinds of network e.g. an agent having a wireless internet connection as well as an infrared communication link). The control logic of the control unit should be based on some fixed values (thresholds, limits or any other values used in the branching logic of the control unit), let us call those values collectively as control parameters C[.](t) at any particular time t. Now if we have an agent in which these control parameters are the same for all instances of time, we call it a passive agent. Note that a passive agent may still be a smart piece of software, but it doesn't has the "capacity" (or to be more precise "necessity") to learn and modify its control logic with time. Looking at "Fig. 4", a passive agent will just contain the components painted white (Control Unit and Control Parameters).

B. *Active Agents and Activity*

In previous section we saw that the control unit in a passive agent is taking decisions based on the inputs it is getting from its host (base software and host hardware) and from associated agents. The decisions however are governed by control parameters which are not changing with time. In an active agent, these control parameters are updated regularly to modify control unit's working logic. To implement this, the control unit could be storing its logs (some or all inputs, outputs and current control parameters) onto a storage device from where an update mechanism reads some or all logs to generate the new control parameters. In "Fig. 4" the components painted gold are the implementation of activity in the agent. The update mechanism could house any of the AI (Artificial Intelligence) based learning algorithms, or any simpler piece of code that learns from control unit's history and updates the control parameters. In essence, an active agent modifies its behavior with time based on what it learned from the history of events that took place during its runtime.

C. *Hyperactive Agents and Hyperactivity*

In previous section we saw that Activity is an internal phenomenon which enables modification in an agent's working logic with time. Hyperactivity however is a phenomenon at the Multi Agent System level where agents are able to modify the working logics of its associated agents based on their own history of events. Although associated agents in a multi agent system do influence the behavior of other agents by transfer of information through the passive channel, but the novel idea here is in having a separate mechanism for communicating what one agent learned from its history of events, to the learning mechanism of another agent.

In "Fig. 4" the components painted blue are the implementation of hyperactivity in the multi agent system. The hyperactivity mechanism in a hyperactive agent reads from the storage just like the update mechanism, but has a different learning algorithm that generates the new hyperactive outputs (Ho(t+1)) for the associated agent (Agent B in “Fig. 4”). The hyperactive outputs from agent A are the hyperactive inputs to the hyperactivity mechanism of agent B. Similarly, the hyperactivity mechanism of agent C sends hyperactive inputs to agent A, which are placed into the storage with the hyperactive outputs that agent A generated (The Blue components of the logs in the Storage are the hyperactive components which are added by the hyperactive mechanism just like the other logs are added by the control unit “Fig. 4”).

The update mechanism in a hyperactive agent reads the logs generated by both control unit (t, C, i, I, o and O) and by the hyperactive mechanism (Hi and Ho) “Fig. 4” and thus the updated control parameters are directly influenced by the internal events as well as by the events happening elsewhere in associated agents. The transfer of hyperactive updates from one to another agent makes one hyperactive link and a multi agent system with one or more hyperactive links is called a hyperactive multi agent system. An agent needs to be an active agent first (presence of an update mechanism) in order to receive hyperactive inputs from other agents. In a unique case, a passive agent could send hyperactive inputs to other agents provided it has storage and hyperactivity mechanism (and no update mechanism as it’s a passive agent).

IV. The Hyperactivity View

In section II we discussed the Hyperactive Transaction Model and its first layer based on Agent Relation Charts [11]. We have seen that the hyperactive transaction model has hyperactivity view as one of its views which capture the hyperactivity related design components in the model. In agent relation charts, we have a Hyperactivity ARC (HARC) in the hyperactivity view which is used to specify which of the agents in the multi agent system are Active or hyperactive. HARC also shows all hyperactive links in the multi agent system and thus provides a top layer (ARCs are in the topmost abstract layer of the Hyperactive Transaction Model) description of the hyperactivity in the system. “Fig. 5” shows an example [11] of the HARC where system’s hyperactive links are specified and the active agents carry a blue star.

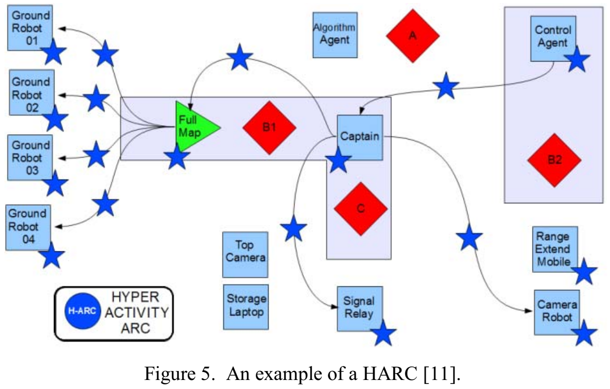

Figure 5. An example of a HARC [11].

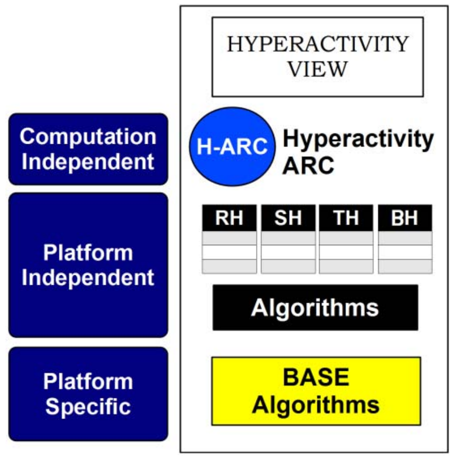

Figure 6. Components of Hyperactive View.

In lower layers of the Model (Under development), the hyperactive view contains the hyperactive components of Platform specific and Platform Independent design of individual agents. It is achieved by having a hyperactivity sub-view (RH, SH, TH and BH in “Fig. 6”) for all other views (structural, relational, trade and behavioral views). The hyperactive part of the relational, structural, trade and behavioral logic is placed in these sub-views for both middle layer platform independent design and for platform specific base software design at lowest layer.

The communication of information generated by individual agents based on their event history, to other agents in the multi agent system is an important aspect of an intelligent multi agent system. Thus, hyperactivity is an important view (views are earlier discussed in “Sections I.D” and “Section II”) for the model. This separation of hyperactivity from other views is the key element of the Hyperactive Transaction Model as it enables independent handling and modeling on system hyperactivity.

V. Discussion and Future Work

In the present paper we first presented a background introduction to software engineering for multi agent systems, model based engineering and multi view modeling. Hyperactive Transaction is a multi-view model for modeling multi agent systems. Agent Relation charts are modeling tools for the top layer of this model which has five views. These views capture different aspects of the multi agent system and provide a clear method to model them separately. The hyperactivity view of this model is unique as is is based on a new concept. In this paper we have tried to give an introduction to this concept and explained the importance of this view for the hyperactive transaction model. In future, the lower layers of the model will define the platform independent and platform specific layers of the model. The authors believe that a multi-view approach with hyperactivity as the base concept will enable system designers and developers to model and implement complicated systems with ease. The designs made in Hyperactive Transaction Model will be easily translated into UML, which is the most commonly used design description language used in the industry. A number of case studies are planned to justify the usability of the model and will provide useful insight to further refine the model.

VI. Acknowledgment

The current research is being funded by the Laboratory Le2i (CNRS 5158, Le-Creusot, FRANCE), Bourgogne regional council (Regional French administration) and the University Technology Petronas (Perak, Malaysia).

References

[1] M. Luck, P. McBurney and C. Preist. "Agent Technology: Enabling Next Generation Computing", In A Roadmap for Agent-Based Computing, ISBN: 0854327886, ver. 1.0. Southampton: AgentLink 2003.

[2] M. Luck, P. McBurne, O. Shehory and S. Willmott. "Agent Technology: Computing as Interaction", In A Roadmap for Agent Based Computing, Compiled, written and edited by M. Luck, P. McBurney, O. Shehory, S. Willmott and the AgentLink Community, pp. 11-12, 2005.

[3] M. Wooldridge. "An Introduction to Multi-agent Systems", Published in February 2002 by John Wiley & Sons (Chichester, England), ISBN: 0 47149691X, 2002.

[4] N. R. Jennings and S. Bussmann. "Agent-Based Control Systems. Why Are They Suited to Engineering Complex Systems?", In IEEE Control Systems Magazine, vol. 23, no. 3, pp. 61-73, Jun. 2003.

[5] P. Stone and M. Veloso. "Multi-agent Systems: A Survey from a Machine Learning Perspective", In Autonomous Robots, vol. 8, no. 3, pp. 345-383, July 2000.

[6] L. Panait and S. Luke. "Cooperative Multi-Agent Learning: The State of the Art", In Autonomous Agents and Multi-Agent Systems, Ed. Springer-Verlag, vol. 11, no. 3, pp. 387-434, 2005.

[7] G. Weiss. "Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence", Edited by Gerard Weiss. ISBN: 0-262-23203-0, 1999.

[8] Diego ALONSO, Cristina VICENTE-CHICOTE, Francisco ORTIZ, Juan PASTOR, Bárbara ÁLVAREZ "V3CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software Development," Journal of Software Engineering for Robotics (JOSER), January 2010, 3-17.

[9] OMG, Model Driven Architecture Guide, version v1.0.1, omg/2003-06-01, Jun. 2003. http://www.omg.org/docs/omg/03-06-01.pdf

[10] E. Seidewitz, "What models mean", IEEE Softw., vol. 20, no. 5, pp.26–32, 2003.

[11] V. Nagrath, F. Meriaudeau, A. Saeed Malik, and O. Morel, "Agent Relation Charts (ARCs) for Modeling Cloud based transactions," in Proceedings of the International Conference on Communication Systems and Network Technologies. Rajkot, India: Laboratoire Le2i, UMR CNRS 5158, IUT, Le Creusot, FRANCE; University Technology Petronas, Perak, MALAYSIA, 2012.