CGP-Net to model Multi-Agent-Systems

Jean-Dany Vally
MAS² IREMIA, Université de la Réunion,
15 Avenue René Cassin - BP 7151,
97715 Saint-Denis messag cedex 9,
La Réunion, FRANCE,
jdvally@univ-reunion.fr

Rémy Courdier
MAS² IREMIA, Université de la Réunion,
15 Avenue René Cassin - BP 7151,
97715 Saint-Denis messag cedex 9,
La Réunion, FRANCE,
courdier@univ-reunion.fr

ABSTRACT
In this paper, we present an uniform way to model agents and multi-agents systems (Mas). The modeling we introduce relies on a simple and powerful basic model: the CGP-Net. This model is an hybrid model binding Conceptual Graphs (CGs) and Colored Petri Nets (CP-Nets). The agent’s particular features are simply modeled with the CGP-Net which intrinsically offers advantages to handle and interrogate the modeled systems. Indeed, one major drawback to the Mas design lies in the difficulty to express in a generic way the possibility to extract precise informations from a running system. In the existing platforms the use of static tools (mainly for observation) compromises in part the user’s needs. The CGP-Net brings naturally a solution to these expectations. And its integration in the generic and flexible Mas model gives several advantages for the system’s exploitation as well as for the agent’s and Mas’s design.

Keywords
Multi-agent Simulation, Evolution of Agents, Conceptual Graphs, Colored Petri Nets

1. INTRODUCTION
The multi-agent paradigm has been successfully applied to the development of simulation environments. We have proposed a generic simulation platform [MG08] (GENeric Architecture for Multi-Agent Simulation) designed to support specific considerations of non-linear complex systems as defined in [BG06]. Our aim is now to extend the scope of applications supported by Geamas by accepting applications that are either reactive or cognitive and to propose methodology principles to guide the process of the system design [CM08].

The originality and all the difficulty of our process lies in the research for a proposal of a methodology adapted to the complete MAS life cycle. During the design stage, this also implies the consideration of the static and dynamic aspects of entities, as the consideration of agents handling during the process. This approach is the result of the report that it is very difficult to understand what occurs in a MAS made up of hundreds of autonomous and self-adapting agents without integrating, since the design stage, some modeling tools adapted to their observation.

To establish our reflection foundations, we already have powerful formal elements used in object-oriented methodologies [RBP 91], in agent-oriented methodologies [CPD96] [Bur96] and in fields like SGBD [BM91] (for the querying of more complex systems).

In other fields such as linguistic, the semantic structures (Conceptual Graphs [Sow84], …) express knowledge in a logically precise form, humanly understandable, and computationally tractable. The Conceptual Graphs (CGs) model is rich enough to include the main features of the object-oriented model [Sow93] [EB95] and to encompass new directions in AI. Moreover it provides some well suited forms to support useful querying aspects. But it shows some lacks when we have interest in efficient representation of dynamic process. Colored Petri Nets (CPNs) [Jen92] [Jen88] is a graphical oriented language for design, specification, simulation and verification of dynamic systems. Such model completes in a fashionable way the lacks of the CGs model, and does not require to modify profoundly both models with artificial concepts. Our work consists in providing formal elements useful for the modeling of multi-agents systems properties. Such elements can be derived from existing formal elements. We think that the conceptual graphs and colored petri nets, which have not been both evaluated for the modeling of multi-agents systems, offer a good base for the design of conceptual modeling tools that support the agent-oriented methodological process that we want to work out of Tab. 1.

The article is organized in four sections. Within the first one, we present and discuss methodologies and models concerning the agents and multi-agents systems. After this description, we specify the features of our generic design model (Agent and Mas). In the next part, we recall the characteristics of the CGP-Net model. And finally, we will discuss the advantages and contributions of our proposal for the agent design and Mas observation. In conclusion, we present future working lines.
2. FEATURES OF AGENT’S METHODOLOGIES AND MODELS

After the fashion of object-oriented methodologies [RBP ’91], some notions repeat: organizations, groups and roles. In the different methodologies these notions are more or less formalized with more or less close semantics.

In the Agent-Oriented Analysis and Design methodology [Bur96], three models are defined for analyzing an agent system: the Agent, Organizational and Cooperation model. In this methodology, the modeling of specific agent features relies on extensions of object models.

The method Agent Modeling Technique for Systems of BDI agents [KGR96] defines two main levels: the external viewpoint for the decomposition of the system into agents and their interactions and the internal viewpoint for the Modeling of each BDI agent class with three models (belief model, goal model and plan model).

One of the first methodologies to appear is the Cassiopeia method [CPD96]. The Cassiopeia method is a way to address a type of problem-solving where collective behaviors are put into operation through a set of agents. The main concepts in Cassiopeia are those of role, agent, dependency and group. An agent is viewed through three levels: Individual, Relational and Organizational role.

The organizational point of view is also present in Aalaadin [FG98]. The main model for Aalaadin is the agent-group-role model: the agent can handle roles within a group, roles are functionalities or services of an agent and the group structure associated with the roles enables to express several organizational types. It is assumed that an agent is an active, communicating entity which plays roles within a group (roles are identified within a group).

Aalaadin or Cassiopeia describes a first control of collective behaviors by the use of graphs. We undertake to go still further in this way by using graphs formalisms adapted to the modeling of the agent paradigm specificities.

Now we have recalled the usual features of multi-agents systems, we can concentrate to formalize in a generic way the more relevant features of our agent paradigm.

3. UNDERSTANDING THE AGENT SPECIFICITIES

Until now, the design work was dedicated to the coupling reduction between components in order to facilitate an iterative construction of the system. The model’s stability compared to the real world entities was increased in a punctual way but without preserving the unification capacity present in the object approach. Some people succeeded in preserving a significant capacity of integration by completely disregarding the agent internal architecture [FG98].

The agent approach is based, as the object approach, on the integration of what the system is and do. If the collective layout of the parts is significant to obtain a more global view, one should not neglect the interest to progress at various abstraction levels. The agent evolution is dual: internal and global. And one must give a detailed attention to the system components.

The purpose of the agent approach relies on the modeling of autonomy and adaptation for the system in which the tasks are defined. This approach enriches the object one’s that is based on the static and dynamic entities integration. The system decomposition into simple elements is done in a similar way, the agent is an iterative abstraction of a real world entity. This abstraction is done as well on the structural properties as on the functional properties (adaptability in a whole) of the entity.

Finally, a Mas results in a flexible integration of independent, autonomous and adaptive entities. To enhance the power of this integration the Mas model is matched with the agent one cf Fig. 1.

Our solution to obtain all these expectations relies on a standardization of the Action/Reaction mechanism to model Sensors/Motors (interface), Access/Storage (Internal memory), Lack/Pro-action (Adaptation) and Request/Results (Observation). The CGP-net model provides an efficient modeling of all these mechanisms.

3.1 Evolutive agent definition

We will rely on an intuitive definition, now traditional, of agent before stating a more practical definition.

**Definition 1:** An agent is an encapsulated computer system, situated in some environment and able to act in a flexible and autonomous way in order to meet its design objectives [Jen00].

**Definition 2:** An agent a is a polymorphic entity which
Different agent’s modes are defined on ontological supports (Signals’ types hierarchies, ...) enabling it to interact with other parts of the environment. These modes are modeled with the CGP-net formalism.

So, the agent is an atomic unit made of union of a structure and modes. It is an exchange support simultaneously getting a large integration capacity in addition to an increased independence with regard to the outside world. The agent becomes a phenomenon revelator when, by the means of information exchanges, it is integrated in a simulation.

**Proposal 1**: An agent is more formally defined by the four following characteristics: a structure, a logic, a fluctuation source and an identity.

The agent structure defines its sensitive characterization in the environment by providing support for an internal and external exchange of Fig 3.

The agent logic is a set of adaptation modes cf Fig. 4. Each mode is a way to interact with the environment while satisfying the agent lacks. A mode is not only the addition of capacities, it also brings a precise semantic information about the agent adaptation in an organized environment.

The fluctuation source describes the handling of lacks. It reveals the agent’s proactive property. This source is dispatched between the different agent’s modes. In the third part we recall our CGP-Net definition and illustrate its use with agent’s modes.

All the agent’s features are useful to model a flexible integration at the system level.

**3.2 Relations with the system (W)**
The agent’s independence with the other parts of the system rises directly. This is due to the fact that the interactions are done by knowledge transfer between SFs without direct calls to behaviors. In counterpart the system, as a whole, must ensure the knowledge transfer between its various parts. SFs and modes enable to model these two interaction types without deteriorating the agent’s independence.

The agent’s interactions as a part of the system are of two types [Var] :

- The **vertical interactions** describing dependencies between the system’s properties and the part’s properties (and vice-versa).

- The **horizontal interactions** describing constraints among parts which characterize the system's integrity.

Moreover, the vertical interactions (which induce the representation of the system as a whole) offer the possibility to “objectify” or “agentify” the signals emitted by the agents, cf Fig. 5.

The modes describing the whole, or the parts, are represented by a model adapted to the specificities of the agent’s paradigm: the CGP-Net model. These specificities intrinsically bring a flexibility for the adaptation and evolution of the system.

### 4. AN HYBRID MODEL

Actually, agents unlike objects are evolutive entities with their own motivations for acting in the world. An agent evolves with a more or less improved cognition.

This cognition can be modeled with different faculties at different improvement levels. Perception, memory, learning, reasoning, understanding, and action are different aspects of the same process of cognition. These aspects were very
4.1 The conceptual graphs

Conceptual graphs are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of IA. Many popular diagrams can be viewed as special cases of conceptual graphs: types hierarchies, entity-relationship diagrams, dataflow diagrams, state transition diagrams and petri nets [Sow84]. Conceptual graphs embed these notations in a general framework of logic. No extensions of the theory or the notation are needed to use conceptual graphs as a design language for object-oriented systems [Sow93] [Ell93].

Some tools have been developed to support a precision information retrieval like WebKb [ME98] which retrieves informations on Web-accessible databases, Notio [SL99] a Java API for constructing conceptual graphs tools, etc.

4.2 Coloured Petri Nets

Coloured Petri Nets [Jen93] is a modeling language developed for systems in which communication, synchronisation and resource sharing play an important role. CP-nets combine the strengths of ordinary Petri nets with the strengths of a high-level programming language.

CP-nets have computer tools supporting their drawing, simulation and formal analysis. Moses [REN97], Renew [OK00], . . . are high-level Petri net simulators that provides a flexible modeling approach. Existing tools for the definition and use of CG and CPN are available and extensible, we can rely on these tools to develop a simulator for Multi-Agents Systems.

4.3 The CGP-Net model

The merging of the two models is done informally in the following way. We directly associate the values of tokens with conceptual graphs. These values will belong to the Conceptual Graph type. Arcs expressions, possibly referenced by variables, will contain conceptual graphs used to identify valid tokens. These conceptual graphs could be bound to variables to represent them in other arcs expressions. The possible operations inside the arcs expressions are combinations of the canonical formation rules of the CGs. And finally, the guard functions are expressions where a particular relation type called actor could be used. We recall the definition from [VC02]

**Definition 3:** A CGP-net is a tuple CGP=(CG,P,T,A,N,G,E,I) where:

- (i) CG is the Conceptual Graph type,
- (ii) P is a finite set of places,
- (iii) T is a finite set of transitions,
- (iv) A is a finite set of arcs such that: P ∩ T = P ∩ A = T ∩ A = ∅,
- (v) N is a node function and is defined from A into P × T ∪ T × P,
- (vi) G is a guard function defined from T into expressions such that: ∀t ∈ T | Type(G(t)) = B ∧ Type(Var(G(t))) = Conceptual Graph,
- (vii) E is an arc expression function defined from A into expressions such that: ∀a ∈ A | Type(E(a)) = Conceptual Graph ∧ Type(Var(E(a))) = Conceptual Graph,
- (viii) I is an initialization function.

(ii) + (iii) + (iv) The places, transitions and arcs are described by three finite sets that are pairwise disjoint.

(v) The node function maps each arc to a couple in which the first element is a source node and the second is the destination node. The two nodes must be of different types (i.e., one of them a place and the other a transition).

(vi) The guard function G, maps each transition t into a boolean expression where all variables belong to the Conceptual Graph type.

(vii) The arc expression function E maps each arc to an expression belongs to the Conceptual Graph type.

The CGP-Net's evolution does not differ from the classic CPN's evolution (binding, step, occurrence) [Jen93]. However, we must notice that in classical CP-Nets arcs expressions are identified values or variables but moreover in our model they are structured knowledge queries. It allows on the one hand to do an abstraction (and thus simplification) of several bindings we have interest in and, on the other hand to increase the expression since the CGP-Net itself can be referenced as a concept in arc expressions.

This simple improvement enables us to model Mas in a generic way, indeed it completes the work done with Types (Is-a relation) by invoking relations between Instances and Types. Is instance of is no more the only Type's relation to manage instances, now with a CGP-Net modeling a Type, we can model Instance mutations. The tokens representing instances are explicit in the Type definition (a CPN model) and they are transmitted between Types by transitions. Later we call these completions of Types: modes. In this article we do not further develop in this way, we just present the modeling of proactivity for an agent instance. We observe that in this way any strategy could be used to model the instance managing. The CP-net will allow the basic manipulations and the CG the knowledge queries.

We consider that performatives as defined in KQML [CL95] are part of the message (represented by a token), and we concentrate on the entity interpretation and management (arc expressions and transitions). It enables to deal with heterogeneous entities, some will parse performatives or sender and others will be only sensitive to the contents or even parts of the contents.
5. MODELING ADVANTAGES
The main advantages at the agent level are proactivity, observation and manipulation.

5.1 CGP-nets and proactivity
Definition 4: Agents are proactive entities, they do not simply react to their environment, and are able to exhibit goals-directed behaviors to initiate actions. [WJ95]

Definition 5: A mode $m$ of an agent $a$ is defined by [VC02]:

- Processing reactions (access, storage, combination) of knowledge generated by the system. These knowledge’s processes enable the agent to efficiently react according to the system.
- Or answers actions to the absence of information generated by the system. These answers enable the agent to effectively act according to its lacks.

The model used to represent the modes is the CGP-nets. It brings a smart representation for modeling of modes. The structural components are modeled with places (sources, destinations) and the dynamic components with transitions. The reactions indicate the transitions having classical arcs of the CP-nets as sources. The actions indicate the transitions having proactive arcs as sources (functionally corresponding to the inhibitor arcs but bringing a different semantics to the transition which generate the action). Lacks correspond to the proactive arcs expressions. Example: Initiation of an interaction between an agent and a taxi driver due to the lack of food Fig. 6.

5.2 Interrogation and handling of the system
The various operations we wish to apply on multi-agents systems are described in dynamic structures. These dynamic structures can be defined for the environment or a particular agent depending on the concepts concerned with the query/handling. For example we can file all foods consumed by an agent in a particular database Fig. 7. Thus, the token containing the Daniel reference will contain also all food which this one will have consumed.

CGs tools make it possible to extract and compose new knowledge for the system whereas CP-nets tools enable to evolve and backup various states of a system. A prototype relying on Notio and Renew is currently implemented.

Some people may think that CG or CP-net are too basic and heavy models for representing Mas system but we do not forget that for now people are trying to define specific relations or structures between types and instances (for representing in an efficient way evolutive entities). Indeed static relation are already well defined between Types, and investigate other relations (like modalities) are not sufficient to determine useful consequences for instances, we have to manage the instances in some less abstracted ways. So a robust instance model with simple modifications can lead us to generic well defined relations. And moreover, in this instance the model intrinsically support simulation and querying aspects which are useful for dealing with a Mas.

6. CONCLUSION AND PERSPECTIVES
In this paper, we have introduced a useful way to model agent and multi-agents systems. This model relies on a standardization of the action/reaction mechanism to express the various characteristics of an agent and Mas. The agent model is formalized with structures and modes. Such model supports the two interactions types between agent and system. The CGP-net model brings several advantages to express adaptation, learning and interrogation in a multi-agents system. With this model the proactivity is formalized in a simple and intuitive way as well as request within a running system. The prospects of this work are interesting for simulation thanks to multi-agents systems since it lies on a reliable basis for which a great deal of work has been already done concerning the formulation of requests. This opening with regard to the possibility for expressing requests very significant in the field of multi-agents simulation. In-
Indeed, with this capacity to observe and manipulate Mas, we expect in our future work to propose some simple ways to efficiently model emergence in Mas.

7. REFERENCES


A. Varzi. Parts, wholes, and part-whole relations: The prospects of mereotopology.
