Towards the Formal Specification and Verification of Multi-agent Based Systems

Boucherit Ammar and Khebaba Abdallah

Computer Science Department, Ferhat Abbas University
Setif, Algeria

Abstract
This article is directed primarily at the problem of developing a more reliable multi-agent based systems, because the paradigm of multi-agent systems, which offers an original way of modeling, is considered as an appropriate method that faces the problem of modeling, and it is present in the most of sectors: telecommunications, finance, Internet, energy, health, embedded systems ... etc. Therefore, it is crucial to have rigorous, automatic and effective design and checking methods to ensure their development.
The main objective of this paper is to present and discuss a new approach for the formal specification and verification of agent based modeling system. In addition, our approach is based on rewriting logic, includes a well-known and effective verification technique, model checking, and allows independent of the used formalism to verify a large set of interesting properties deemed relevant on multi-agent based system.
Keywords: *Specification, Verification, Rewriting Logic, Agent-Based Modeling, Model-Checking.*

1. Introduction

Firstly, the paradigm of multi-agent systems [1, 2], which offers an original way of modeling, is considered as an appropriate method for computer systems. Therefore, multi-agent based modeling method is present in the most of sectors: telecommunications, finance, Internet, energy, health, embedded systems ... etc. In other words, agent-based Modeling is one of the techniques that can be used to model any kind of systems. What distinguishes this approach from others is that it facilitates a m ore direct correspondence between the entities in the target system and the parts of the model that represent them (i.e. the agents) [3]. For example, in a production factory, the behavior of a co mplex machine that has own internal situations, its own rhythm, different reactions in different situations, can be effectively modeled by an agent that can be integrated with the model production chain.

Secondly, even if multi-agent based modeling approach has the potential and the capability to model different systems [4]; these potentialities should not hide the difficulties associated with them in the design. These difficulties may discredit the field of agent based modeling as a whole and affects their relevance, and their scientific credibility. Moreover, at this time there is no evidence of a well-established engineering approach for building multi-agent based applications.

Thirdly, it becomes crucial to have rigorous methods of formal specification and verification to ensure the safe development of agent based systems, which may be critical systems, and not risk erroneous attribution to this type of system, some properties such as security, integrity and robustness. In other words, the need for rational methodologies and tools of assistance to the design of these systems remains major. This assistance should not be limited to the run phase, but it must cover all the process of their development. The use of the formal methods can provide a help very useful for the analysis and the specification of the multi-agents system thanks to the rigor of their methodologies. Moreover, they allow an easier passage to the implementation and facilitate the difficult phases of checking and tests.

The main objective of this work is to contribute first to the establishment of a unifying framework for all the specification models of MAS. The second contribution is to prove some interesting properties for MAS who can be of two types: intrinsic with the systems for example, the communication and co-operation or related to the formalism used for specification.

In order to present our approach, we start by introducing the useful theoretical concepts and difficulties related to multi-agents systems. Then, we specify the studied multi-agents system in the general context of the rewriting logic through its practical environment MAUDE. Finally, a large set of interesting properties of multi-agents systems will be verifyied by using MAUDE's LTL model-checker and Search tool. In fact, this approach is the extension of our previous work [5, 6], in which we have not used the Search tool that enlarge the set of properties can be verified in the studied system.

2. Preliminaries

In this section, we present the scientific context and formalisms to be used in our approach for the formal specification and verification of multi-agent based systems.

2.1 Agent and Multi-Agent Systems

The increasing complexity of the industrial systems and the delocalization of the processing call more and more upon the use of new techniques where the processing can be decentralized. Therefore, this situation imposes the need for using *entities* able to solve problems, and also equipped with capacities of *communication* and *social reasoning*, i.e., they are able to reason the ones on the others. These entities are known with the name of *Agent*. Where an agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives [7], and the set of these agents, with these various capacities constitute *a Multi-Agents System* (MAS).

Various definitions have been proposed for the term multi-agent system and according to [1], "Multi-agent systems are a new paradigm for understanding and building distributed systems, where it is assumed that the computational components are autonomous: able to control their own behavior in the furtherance of their own goals".

The most important reason to use agent paradigm when designing a system, is that agent has the potential to enhance the transparency, reliability and rigor of the modeling process, and because some domains require the competence of a set of agents in order to solve problems, which are difficult or impossible for an individual agent. In other words, the agent-based approach is more akin to reality than other modeling approaches, and in many cases, agent-based modeling is a natural method for describing and simulating a system composed of real-world entities in very realistic ways [53]. The main characteristic of an agent is its autonomy, which provides the capacity of action within the environment in order to achieve its goals. This characteristic gives the agent the aptitude to model complex and real time systems in a distributed way, using a behavior generation approach where different control locations are available. In addition, agent based model has the ability to design heterogeneous and complex system, where agent can represent any type of unit, from which the system can be formed.

Finally, the potentiality of multi-agent system as an original way of modeling should not hide the difficulties associated with them in design and verification. We can summarise the inherent difficulties in three points:

1. The lack of mature of well-established engineering approaches for building MAS-based applications.
2. The agent-based modeling has generated lots of excitement and the absence of proof for general properties of a model leads to problems that may affect multi-agent systems [8].
3. It would be practically impossible to develop universal MAS Library; design generic and secure models especially for safety critical systems.

Therefore, it is important to ask about the validation, and search for rigorous, automated and efficient methods of design and verification for agent-based systems. The disposition of such methods will help the designer to develop, validate and ensure the reliability of multi-agent based system before its implementation.

2.2 Rewriting Logic and MAUDE System

Rewriting logic is a computational logic proposed by Meseguer [9] as a natural model of computation and an expressive semantic framework for concurrency, parallelism, communication and interaction, which builds upon equational logic by extending it with rewrite rules to adapt it to changes [10], and specification of concurrent systems. Rewriting logic can be used for specifying a wide range of systems and languages in various application fields. It also has good properties as a metalogical framework for representing logics.
In rewriting logic each concurrent system from simple to more complex models can be specified easily by the use of, a rewrite theory $T = (\Sigma, E, L, R)$. Its static structure is described by the signature (Σ, E), whereas its dynamic structure is described by the rewriting rules R.
In other words, a rewrite logic theory consists of a set of uninterpreted operations constrained equationally, together with a set of rewrite rules meant to define the concurrent evolution of the defined system. The distinction between equations and rewrite rules is only semantic.

A rewrite theory is a 4-tuple $T = (\Sigma, E, L, R)$, with:

- *(Σ,E)*: an equational theory with function symbols Σ and Σ-equations E;
- *L* set of labels; and
- *R*: a set of labeled rewrite rules of the general form : $t \rightarrow t'$ if C, where t and t' are algebric terms describes a particular structure for the states of a system. The rewrite rules describe which elementary local transitions are possible in the distributed state by concurrent local transformations if a condition C is verified [11].

For any term t in the rewrite theory *T*, we write [t] for its equivalence class, and we say that [t] → [t'] is provable in T when it is obtained by a finite application of the following rules:

1. **Reflexivity**: for each term $[t] \in T_{\Sigma,E}(X)$,

$$\frac{}{[t] \rightarrow [t']}$$

2. **Congruence**: for each operator $f \in \Sigma_n$, $n \in N$

$$\frac{[t_1] \rightarrow [t'_1] \; \ldots \; [t_n] \rightarrow [t'_n]}{[f(t_1, \ldots, t_n)] \rightarrow [f(t'_1, \ldots, t'_n)]}$$

3. **Replacement**: for each rewriting rules:

$r : [t(\overline{x})] \rightarrow [t'(\overline{x})]$ if

$[u_1(\overline{x})] \rightarrow [v_1(\overline{x})] \wedge \ldots \wedge [u_k(\overline{x})] \rightarrow [v_k(\overline{x})]$ in *R*,

with $\overline{x}$ abbreviating $x_1, \ldots, x_n$

$$\frac{[w_1] \rightarrow [w'_1] \ldots [w_n] \rightarrow [w'_n] \quad [u_1(\overline{w}/\overline{x})] \rightarrow [v_1(\overline{w}/\overline{x})] \ldots [u_k(\overline{w}/\overline{x})] \rightarrow [v_k(\overline{w}/\overline{x})]}{[t(\overline{w}/\overline{x})] \rightarrow [t'(\overline{w'}/\overline{x})]}$$

with $\overline{w}/\overline{x}$ indicate the substitutions of x_i by w_i $1 \leq i \leq n$.

4. **Transitivity** :

$$\frac{[t_1] \rightarrow [t_2] \quad [t_2] \rightarrow [t_3]}{[t_1] \rightarrow [t_3]}$$

Deduction Rules of the Rewriting Logic

Several languages based on rewriting logic were created, the most known are: CAFEOBJ (Japan), ELAN (France) and MAUDE (SRI, United States). MAUDE [12] is a high-level language and a high-performance system supporting executable specification and declarative programming. MAUDE is based on rewriting logic where several dynamic and concurrent applications can now be considered. The rewrite theory can describe the system as a configuration of objects declaratively with a high degree of abstraction. MAUDE has been used for specification, prototyping and testing of a wide range of applications, because it h as a collection of formal tools supporting different forms of verification such as:

- The MAUDE Termination Tool : can be used to prove termination of functional Modules;
- The MAUDE Church-Rosser Checker : can be used to check the Church-Rosser property of unconditional functional modules;
- An inductive Theorem Prover : to verify properties (theorems), which are defined in functional modules;
- The MAUDE Coherence Checker : can be used to check the coherence (or ground coherence) of unconditional system modules; and
- The MAUDE Sufficient Completeness Checker: can be used to check that defined functions have been fully defined in terms of constructors.

To clarify the theoretical notions of the rewriting logic and its expressivity, we give the following example:

Example:
Our example consists to express in rewriting logic, the system modeling a vending machine, which is used to buy cakes and apples; the cake costs a d ollar and the apple three quarters. Due to an unfortunate design, the machine only accepts dollars, and it returns a quarter when the user buys an apple; to alleviate in part this problem, the user machine can change four quarters into a dollar.
We can represent graphically such a machine as a Petri net as follows.

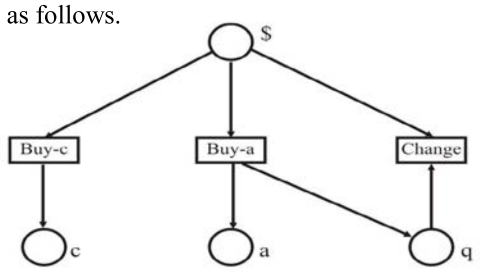

Fig. 1 Vending-Machine Petri net

The static aspect of this system is defined by the vended products (a, c) and the accepted type of money ($, q). The machine can perform the following actions:

1- to sell a cake (vend-c),
2- to sell an apple (vend-a) or
3- to change four quarters into one $ (changes).

Therefore, the state of the machine can be defined by a pair (Z, T) where Z represents the available money in the machine and T defines the product(s) has been sold by this machine. This state can be evolved by executing the possible action. By using MAUDE language syntax, we can write the corresponding module to our example easily:

```
mod Vending-Machine is
sorts  product money stats .
ops apple cake I : -> product .
op __ : product product -> product [assoc comm id: I] .
ops $ q 0 : -> money .
op __ : money money -> money [assoc comm id: 0 ] .
op <_,_> : money product -> stats .
op __ : stats stats -> stats [assoc comm] .
var M : money .
var X : product .
rl [vend-c] : < M $, X > => < M, cake X > .
rl [vend-a] : < M $, X > => < M q, apple X > .
rl [change] : < M q q q q, X > => < M $, X > .
endm
```

2.3 Model-Checking and MAUDE's LTL Module

Formal methods are rigorous techniques based on mathematical notation that can be used to specify and verify software models. A model is defined as a formal representation of the real world [13]. Its purpose is to provide a p icture or an abstract representation of a r eal phenomenon. Model checking is a formal verification technique [14] [15] [16], that determines whether given properties φ of a system are satisfied by a model M. We write this as a judgment M $\models$ φ and say a model checker verifies or refutes such judgments, based on a partial or exhaustive exploration of the state space of the model. The main objective of this technique is to ensure that none of all these states is inconsistent with the desired behavior.

The software tool validating a model and solving the model-checking problem is called model checker. A model-checker typically as presented in the following figure (Fig. 2) supports two different levels of specification:

1) System specification level, in which the concurrent system to be analyzed is formalized; and
2) Property specification level, in which the properties to be model checked -for example, temporal logic formulae- are specified

The model-checker takes these two specifications as inputs, and outputs either a claim that the property is true or a co unter example reporting the inconsistency. A counterexample is an execution trace of the state machine showing how the predicate is false.

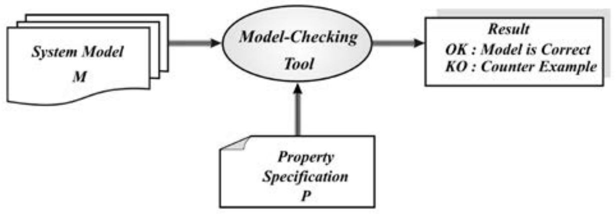

Fig.2 Model-Checking Technique

The MAUDE's LTL model-checker is a v ery powerful model-checker. It was designed with the goal to combine a very expressive and general system specification language (Maude) with an advanced on-the-fly explicit-state LTL model-checking engine. Indeed, to verify such a property, the MAUDE's LTL model-checker takes as inputs the following modules:

1. Rewrite theory specified by a Maude system module M-SYSTEM, which describes the behavior of the system;
2. PROP-M module, which contains the set of predicates expressed in standard LTL propositional logic as the defined syntax in the module SATISFACTION; and
3. The initial state, from which the model checker starts checking, is specified in module M-CHECK.

Next, we have to load the module M-CHECK, in order to check the property formula expressed in linear temporal logic (LTL) with the MAUDE's LTL model-checker.
The main modules used by the Maude's LTL model-checker are presented in the following figure (Fig. 3).

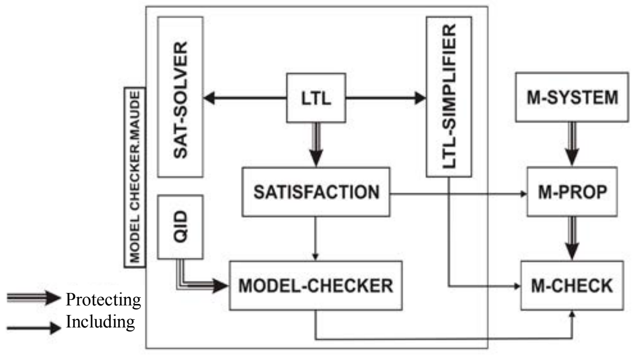

Fig.3 Principal Modules of Maude's LTL Model-Checker

The MAUDE's LTL model-checker modules have well defined roles as follow:

- MODEL-CHECKER: This is the main module in the verification process.
- LTL: This functional module formalizes the syntactic and semantic definitions of linear temporal logic.
- LTL SIMPLIFIER: It tries to further simplify the negative normal form of the formula ¬φ : in the hope of generating a smaller Büchi automaton B¬φ.
- SAT-SOLVAR: It can be used to check both satisfiability of an LTL formula and LTL tautologies.
- SATISFACTION: A very simple module defines the standard LTL propositional logic used to express the set of predicates.

3. Multi-Agent Based Systems Formalization

Formalization is the process to transform a less-formal system or model into a more-formal one. In this section, we need to present as possible all related points to this area of research in order to take a good idea about its different varieties, and to reveal clearly where the lack is situated.

3.1 Multi-Agent Systems Formalisms

Various formalisms were used to formally describe distributed systems based on agents. These attempts made it possible to associate a f ormal semantics the modeled

systems and to have a clear expression of their properties, as well as a co herent and transparent vision of what the made system do or does not do. Nevertheless, the complexity of the multi-agents systems and their dynamics make their formal checking too difficult.
As what we have noted above, we can find in the literature several attempts for the formal specification of multi-agent systems, which tend to describe an agent in mathematical terms, and those based on Petri nets, finite state automata, X-machine such as :[17, 18, 19, 20, 21, 22, 23, 24], … etc.
In our opinion, the formalisms used for the formalization of multi-agents system can be classified into five principal families:

• **Petri Net:** is one of the powerful formalisms for the description and analysis of concurrent processes, which arise in systems with many components (distributed systems). Since execution of Petri net is non-deterministic when multiple transitions are enabled at the same time. Petri net are well suited for modeling the concurrent behavior of distributed systems. In the context of the development of multi-agent systems, we can find the Petri net formalism and all its extensions [25] [26] [27], which are generally used to describe the internal and external behavior of agent and multi-agent system. Unlike other formalism such as UML, Petri net have an exact mathematical definition of their execution semantics, with a well-developed mathematical theory for process analysis.

• **Logics:** In this family of formalism, we can easily find three big kinds. "Temporal logic" is a s pecial area in the study of logic. It concentrates on studying, representing, and reasoning about system activities (states) with the use of a time line. The "specification logic" is a temporal logic augmented with a knowledge operator. It is used to specify computation. While these logics tend to give an abstract, timed and structural specification of the multi-agent system, "Logic of BDI" divides the mental state of an agent into Beliefs, Desires, and Intentions.

• **UML and Languages:** The main used language in this family is UML, which unify the design principles of a collection of object-oriented methodologies into a single, standard, language that can be easily applied across the board for all object-oriented systems. Some extension of UML such as AML and AUML, which are designed to capture the aspects of multi-agent systems. We can find in the literature other languages such as CASL, which allows the specifier to view agent as entity of mental states, and enables to define from them its behavior [28, 34].

• **Formal Approaches:** The fourth family is reserved for the formal approaches for agent verification and modeling. The purpose of these approaches is to provide a powerful tool for academics and practitioners to design, analyze, test and validate the multi-agent system especially for the case of non-tolerant system. While some of these approaches consider the MAS as a whole, the others focus on agents and their interactions as a central topic. Generally, all the approaches that we have explored give firstly the formal description of roles, relations and interactions to achieve certain coordination in the multi-agent system, in order to ensure some conditions and properties.

• **Other Approaches**: Other attempts and approaches were used for agent-oriented analysis and design. We note the Gaia methodology as an example, which is both general, in that it is applicable to a wide range of multi-agent systems, and comprehensive, in that it d eals with both the macro-level (societal) and the micro-level (agent) aspects of systems. GAIA is founded on the view of a multi-agent system as a computational organization consisting of various interacting roles.

The following figure (Fig.4) presents non-exhaustive list for the attempts were found in the literature for the formalization of multi-agent systems and the used formalisms.

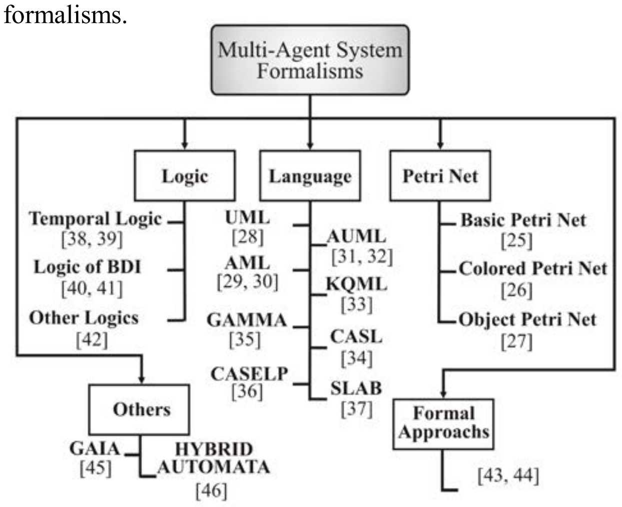

Fig 4. Most Used Formalisms for the Formalization of MAS

In the next subsections, we will present the works that we are seeing significant in the field of specification of multi-agent systems.

3.2 Formal Specification

The process of development of the information processing systems includes a whole of phases such as specification, design, validation and tests. We generally start from an abstract description of the system, using the natural language and the passage to the design phase is intuitive. Nevertheless, when the reliability of the system is too important, it becomes necessary to start from a formal specification, which describes the system behavior by means of a formal language. Many works exist in literature using different formalisms such as Petri nets, Logics, Languages, UML. In general, we can distinguish two major kinds of approaches: [31, 32, 37, 47]:

- Behavioral Approach

The first approach consists in specifying a system by giving a description whose semantics is founded on transition system (operational semantics). This approach makes it possible to describe the behavior of a system like the composition of elementary behaviors. Petri nets, graphs of states, algebras of process and the languages such as ESTELLE, LOTOS or SDL [47, 49].

- Logical Approach

The second approach is generally based on the use of a language making it possible to express the whole of the system properties. In this case, the used language is of declarative type and the system specification will be expressed by a whole of properties using logic formulas. Temporal logics are examples of languages used by this approach for the expression of properties [50, 51, 52].

3.3 Formal Verification

According to the formalism used to represent the system specification, we distinguish two verification approaches: the behavioral and the logic verification. In the first approach, labeled transition systems is the most widely used formalism for the specification, and the verification process of a s ystem property reduces to compare two labeled transition systems S and P. While the second approach, which is generally based on temporal logic to express all the system properties, the decision about the satisfaction of a property formula will be based on model-checking algorithms [5, 46, 48]. The advantage of formal techniques is that they allow drawing more conclusions from specifications in an automated way. In addition, they allow proving complex properties of the system and the domain, or the execution and animation of specifications to provide a semi-automatic correction method of incorrect specifications.

3.4 Synthesis

First, in our opinion, the two specification approaches are complementary, and their combination can be very interesting:

a) The main purpose of the specification is to provide a complete description of the system. This specification must sometimes be described in two different point of views to cover the static (structural) and dynamic (behavior) of the system. Static view provides an overview of the system that is structural while the dynamic view shows the behaviors, interactions and evolution of the system.

b) It is possible to establish (make) another classification with other criterions, for example: a classification based on aspects or kind of properties to be checked (functional and non-functional) of the system. In addition, it is possible that two formalisms that do n ot belong to the same approach in the mentioned classification can be found together in other approaches if we change the classification criterions.

c) The same formalism can be used to model the two aspects of the same system, taking the example of UML static diagrams and dynamic diagrams. Therefore, the same formalism may belong to two different approaches.

Then, formal verification approaches are too important, especially for systems, which are situated in safety critical environments. Thus, it make possible to prove desirable features as well as the absence of unwanted properties for the modeled systems. In addition, the combined analysis of static and dynamic aspects of the studied systems is also necessary for detecting their hot spots.

Finally, the need for rational methodologies and tools of assistance to ensure the design of multi-agent systems remains major because of their different use in practice. This assistance should not be limited to the run phase, but it must cover all the process of their development. The use of the formal methods can provide a help very useful for the specification and the analysis of the multi-agents system thanks to their mathematical rigor.

4. Formal Specification and Verification Approach for Multi-Agent Based Systems

Sometimes we need to test our system to understand how it works, especially when it depends to human life or critical environment system. In such cases, we turn away from the real world to the virtual world of modeling, and experience in a risk-free environment with the proposed model of the system, to find the real solution.

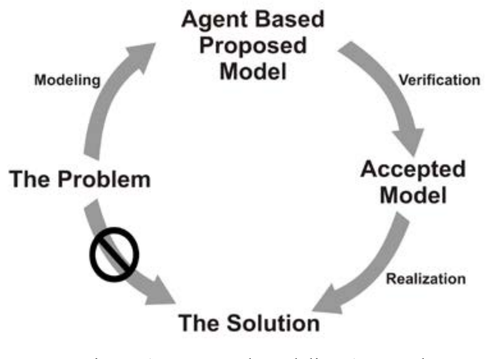

Fig 5. Agent Based Modeling Approach

In this section, we propose a general view of our approach for the formal development of agent based systems.

4.1 Global Description

We illustrate by the following figure Fig.6 the steps of the proposed approach.

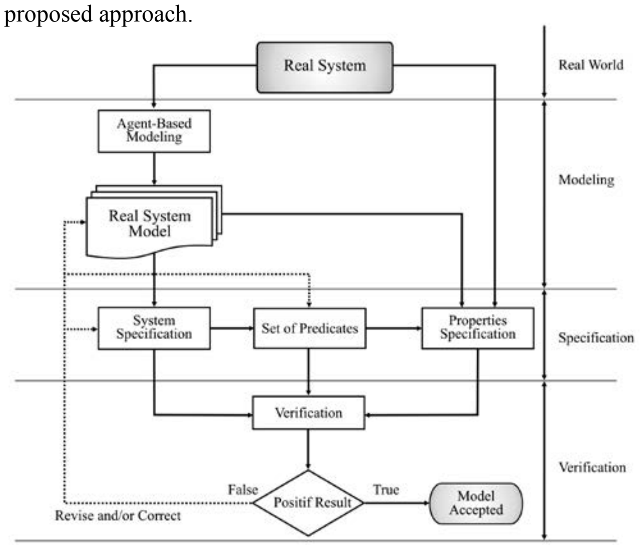

Fig.6. Global Description of the Formal Development Approach

In our approach, which is based on the use of formal and automatic techniques, we start after analyzing system from an agent based model. Then, we transform this model into a specification written in rewriting logic, and we write the specification of the expected properties of the system. Finally, we use the MAUDE's LTL Model-Checker to determine whether the system model satisfies the properties in all its possible executions.

Generally, the steps of specification and verification are essential in designing systems, to avoid any type of error and validate systems before their implementations. Therefore, in the case of agent based modeling may be used in safety-critical cases, the proposed method must be funded on formal specification and verification and it must cover all the process of development, in order to prove the safety of models intended to represent the relevant functions of the studied system.

4.2 Steps of the Proposed Approach

Our approach consists of three phases:

- Modeling

The goal of this phase is to create the first complete version of model for each important property in the system. For that, if the system is truly dynamic or complex, which means that its status changes over time, as it can not be represented by analytical calculations using formulas. The only way left to explore the behavior of the system is to use the agent based modeling.
The first thing to do in this phase is to transform the real system into multi-agent system. Once the relevant agents are identified, the real system can be modeled with different points of view (static or dynamic). Then, according to the property or aspect to be verified, the multi-agent system must be formalized using for each property one of the most relevant formalisms such as UML, Petri nets, labeled transition systems … etc. In addition, it is judicious to consider several models of the same system, in order to profit from the advantages of each one of them and to check a significant number of its properties [54, 55]. This step can be repeated several times until all the interesting actions in the system are well represented. At the end of this phase, a m odel of high quality for each property to be checked is created.

- Specification

The following figure Fig.7, present the detailed modeling and specification phases.

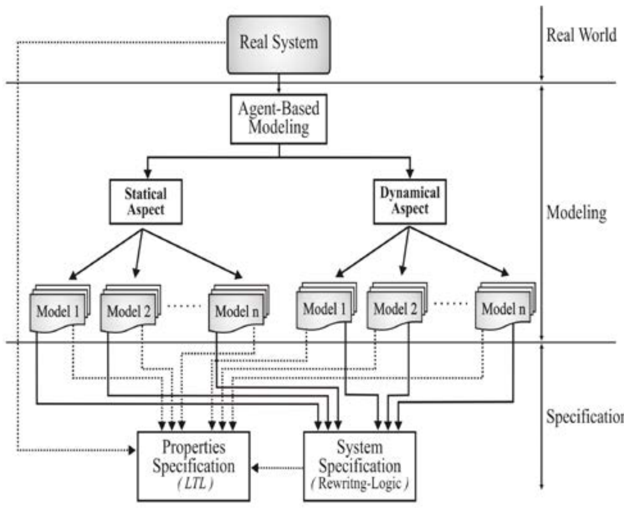

Fig.7 Detailed Modeling and specification phases

In this phase, we have two specification levels:

1. System Specification: The main objective of this specification level is to transcribe (or to give the full description of) the created models in the previous phase, and to express all its expected features in rewriting logic. In other words, the model in this case is transcribed into a set of rewriting rules specifying how to achieve the future state of the system from the current state. Following the nature of the studied system and property to be checked, this combination of rules can be either differential equations, state diagrams (state charts), process diagrams, etc. This stage ends with a description of each model in

rewriting logic, which is logic of change and a unifying semantic framework.

2. Properties Specification: If the aim of system specification level, is to give a more or less abstract description of the system. A system can be formally defined by its properties. In this step, we refer to the created model and its specification to prepare a module that defines the set of predicates expressed in standard LTL propositional logic. These predicates will be considered by the MAUDE's model-checker tool as the set of verified properties in the system. Then, the set of properties to be checked must be also expressed by using linear temporal logic.

- Verification

Finally, the verification step is necessary to show that the system satisfies the desired property and that it exhibits a stable behavior, and/or certify that the probable malfunctions of the system causes only moderate damages. Two verification techniques as illustrated in the figure Fig.8, are applied to perform this step:

- *Model-Checking* : In this technique, we try to check the intrinsic properties of a model by expressing it using linear temporal logic. The verification process is achieved with Maude's LTL model- checker tool.

- *Empirical Test :* This time, we use the MAUDE's Search technique. Its use is based on situations and empirical cases offered by experts in the field; in order to ensure the absence of critical situations in the model. The use of this technique is intended to accomplish the lack of the first technique, which permit to ensure only the properties expressed in linear temporal logic.

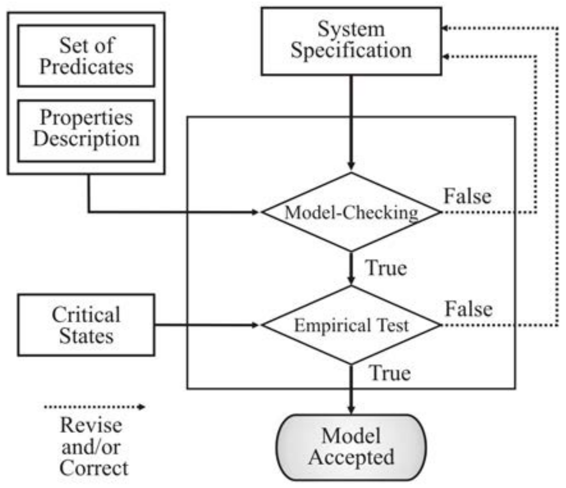

Fig 8. Description of the Verification Steps

4.3 Discussion and Motivation

Firstly, multi-agent systems paradigm offers an original way of modeling, and it is considered as an appropriate method that faces the problem of modeling of complex and critical systems [56]. Nevertheless, there is still some controversy about its use, because there is not enough established methodological practice for incorporating modeling results into true scientific discourse. (Sec 2.1).

Secondly, we transform all the proposed models for the system of study into MAUDE specification by using rewrite theory. Intuitively, the signature (Σ, E) of the rewrite theory T = (Σ, E, L, R) describes the static structure of system, and rewrite rules describe the possible local transitions in the state of the distributed system by concurrent local transformations (dynamic structure). In addition, rewriting logic is known as a flexible logic and as a unifying semantic framework in which other logics and a very wide range of concurrency models and programming languages can be represented, such us : Petri Net [57], Labeled Transition Systems [11], E-LOTOS [58], CCS [59, 60], PLAN [61], Pi-Calculus [62] … etc.

Thirdly, only formal specification is insufficient to ensure the run-time stability, reliability and safety of a multi agent system if it is not followed by verification step. Such approach for the formal development of multi agent systems must include or support formal verification tool. The MAUDE's LTL tool is a very powerful model checker based on one of the most commonly used approach "on-the-fly". The big advantage of the on-the-fly approach to model checking is that, depending on the formula, only a fragment of the overall state space might need to be generated and analyzed in order to produce the correct result (cf. [63][64][65]). Contrary to the classical methods, their effectiveness has been demonstrated, and they were used to analyze real systems of significant size. (cf. [66][67]). Moreover, Maude has a large collection of formal tools (see. Sec 2.2) supporting different forms of verification.

5. Conclusion and Future Work

Agent-based modeling is a very powerful modeling technique that has seen a number of applications in the last two decades, including applications to real-world problems. Therefore, it is becoming increasingly important to make them accessible to formal verification, especially for systems, which are situated in safety critical environments. Thus, it is possible to prove desirable features as well as the absence of unwanted properties for the modeled systems.

In this paper, we have explained our approach based on rewriting logic for the formal specification and verification of multi agent based systems, including well-known formal verification technique (model checking) and the technique of empirical test. Our approach allows to verify a l arge number of properties of a critical system regardless of the formalism used for the specification.

The first advantage of this method is that it is applicable regardless of the type of formalism chosen. In addition, it has the advantage that it permits to verify several types of properties: properties that are expressed and those are not expressible in linear temporal logic. Then, the integration of verification into the design process can detect an error once it occurs and avoids redoing all the verification process by reusing intermediate results.

Our approach still suffers from the problem that it requires a mastery and competence in the use of the formalism of rewriting logic. Because the direct description of a model or the mapping from model to rewriting logic is not always easy.

Finally, in order to palliate this problem in our approach, we intend to continue our research on the development of a framework for the automatic generation of the specification in rewriting logic; at least from the most used formalisms.

6. ACKNOWLEDGMENTS

Our thanks are addressed to all member of LIRE Laboratory for their precious remarks, helps and their contribution to preparing this work.

References

[1] M. Wooldridge. An Introduction to Multi-Agent Systems - Second Edition, Published May 2009 by John Wiley & Sons. ISBN-10: 0470519460.

[2] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, 10(2), 1995.

[3] B. Edmonds. The Use of Models - making MABS actually work. In Moss S and Davidsson P (Eds.) Multi-Agent-Based Simulation, Lecture Notes in Artificial Intelligence 1979: 15-32. Berlin: Springer-Verlag. 2001.

[4] A. Ligtenberg, R.J.A. Lammeren, A.K. Bregt and J. M. Beulens, Validation of an agent-based model for spatial planning: A role-playing approach . Computers, Environment and Urban Systems, Volume 34, Issue 5, August 2010

[5] F. Belala, A. Boucherit. Contribution to the Formal Checking of Multi-Agents Systems. Proceedings of the IEEE International Conference on C omputer Systems and Applications, ISBN: 1-4244-0211-5, 2006, pp. 9-16.

[6] F. Belala, A. Boucherit. Towards a V ideoconference Interface Formalisation, The 4th International Arab Conference on computer science and Information Technology, CSIT06, 2006.

[7] N. R. Jennings. On Agent Based Software Engineering. Artificial Intelligence. Vol. 117, 2000, p. 277-296.

[8] C. Lobry, H. Elmoznino. Combinatorial Properties of Some Cellular Automata Related to the Mosaic Cycle Concept, Acta Biotheoretica, Volume 48, Issue 3 - 4, Dec 2000, pp 219 - 242.

[9] J. Meseguer, Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science, pp 73-155,1992

[10] J. Meseguer, Conditional rewriting logic as a unified model of concurrency, technical report SRI CSL 91, 1991

[11] J. Meseguer, Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science, 1992, pp 73–155.

[12] M. Clavel. Strategies and User Interfaces in Maude at Work. WRS 2003, 3rd International Workshop on R eduction Strategies in Rewriting and Programming - Final Proceedings. Volume 86, Issue 4, December 2003, pp 570-592.

[13] A. Pavé, Modélisation en biologie et en écologie, 560 p. ALEAS Ed, Lyon, 1994

[14] M. Vardi and P. Wolper, An automata-theoretic approach to automatic program verification. In Proceedings of the 1st IEEE Symp. Logic in Computer Science (LICS'86), Cambridge, MA, USA, pp 332-344, June (1986)

[15] O. Lichtenstein and A. Pnueli, Checking that finite state concurrent programs satisfy their linear specification. In Proceedings of the 12th ACM Symp. Principles of Programming Languages (POPL'85), New Orleans, LA, USA, pp 97-107 (1985)

[16] E. M. Clarke, E. A. Emerson and A. P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications. In ACM Transactions on Programming Languages and Systems, volume 8, pp 244-263, April 1986

[17] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. Artificial Intelligence, AI, 42(2-3):213-261, March 1990.

[18] Woolridge M. Temporal belief logics for modeling artificial intelligence systems. Foundations of distributed artificial intelligence. Wiley-Interscience, 1996.

[19] A. Haddadi. Communication and Cooperation in Agent Systems. Lecture Notes in Artificial Intelligence, 1996.

[20] M. Benerecetti, F. Giunchiglia, and L. Serafini. Model checking multiagent systems. Journal of Logic and Computation, 8(3):401 423, June 1998.

[21] Wooldridge. M. Reasoning about Rational Agents. Intelligent Robots and Autonomous Agents. The MIT Press, Cambridge, Massachusetts, 2000.

[22] A. Lomuscio and M. Sergot. On multi-agent systems specification via deontic logic. In J.-J Meyer editor, Proceedings of ATAL 2001. Springer Verlag, July 2001.

[23] A. Lomuscio and M. Sergot. The bit transmission problem revisited. Technical Report 4/2002, department of computing, Imperial College, London SW7 2BZ, UK, 2002.

[24] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. Logic Journal of the IGPL, 11(2):133-157, March 2003.
[25] H. Xu and S. M. Shatz, "An Agent-Based Petri Net Model with Application to Seller/Buyer Design in Electronic Commerce," Proceedings of the IEEE 5th International Symposium on Autonomous Decentralized Systems (ISADS), Dallas, Texas, March 2001, pp. 11-18.
[26] D. Moldt and F. Wienberg, Multi-agent systems based on coloured Petri nets, in *Application and Theory of Petri Nets 1997*, eds. P. Azema and G. Balbo (Springer, Berlin, 1997) pp. 82-101.
[27] Aihua Ren, Hui Jiao, Yunfeng Sun: Modeling Mobile Agent with Object-Oriented Petri Net. ACTA AERONAUTICA ET ASTRONAUTICA SINICA. Vol.24 No.1 (Sum No.182) (2003), pp. 57-61.
[28] D.S. Dillon, T.S. Dillon, and E. Chang, "Using UML 2.1 to model. Multi- Agent Systems", Proceedings of the 6th IFIP Workshop on. Software Technologies for Future Embedded and Ubiquitous Systems,. Italy, 2008.
[29] R. Cervenka, I. Trencanský, M. Calisti: Modeling Social Aspects of Multi-Agent Systems: The AML Approach. AOSE 2005. pp. 28-39.
[30] I. Trencansky and R. Cervenka, Agent Modeling Language (AML): A comprehensive approach to modeling MAS, Informatica 29(4) 2005 391-400.
[31] B. Bauer, J. P. Muller, J. Odell. Agent UML: A Formalism for Specifying Multiagent Interaction. Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer, Berlin, pp. 91-103, 2001.
[32] L. Kahloul, K. Barkaoui, Z. Sahnoun, Using AUML to derive formal modeling agents interactions, aiccsa, pp.109-vii, ACS/IEEE 2005 International Conference on Computer Systems and Applications (AICCSA'05), 2005.
[33] Finin, T., Labrou, Y.: KQML as an agent communication language. In J.M. Bradshaw (ed.), Software Agents, MIT Press, Cambridge, MA, (1997), 291-316.
[34] S. Shapiro, Y. Lespérance, an d H.J. Levesque. The cognitive agents specification language and verification environment for multiagent systems, in Proc. AAMAS, 2002, pp.19-26.
[35] H. Lin and C. Yang, C. Spécification de systèmes multi-agent dans le langage Gamma. Proceedings of the IEEE 19th Annual Canadian Conference on E lectrical and Computer Engineering (CCECE05). Ottawa, Ontario, Canada. Du 7 a u 10 m ai 2006. Numéro de publication du CNRC : NRC 48476.
[36] M. Martelli, V. Mascardi, Floriano Zini. Specification and Simulation of Multi-Agent Systems in CaseLP. APPIA-GULP-PRODE'1999. pp.13-28.
[37] H. Zhu, SLABS: A Formal Specification Language for Agent-Based Systems, International Journal of Software Engineering and Knowledge Engineering, Vol. 11. No. 5, pp. 529-558.
[38] M. Wooldridge. Temporal belief logics for modeling artificial intelligence systems. Foundations of distributed artificial intelligence. Wiley-Interscience, 1996.
[39] M. Fisher, 1996. An introduction to executable temporal logics. Knowledge Engineering Review, 11(1). pp. 43–56.
[40] D. Kinny, M. Georgeff, and A. Rao, "A Methodology and Modeling Technique for Systems of BDI Agents," Tech. Rep. 58, Australian Artificial Intelligence Institute, Melbourne, Australia, Jan. 1996.
[41] M. Wooldridge, "A logic for BDI planning agents" In Pierre-Yves Schobbens, editor, Working Notes of 2nd ModelAge Workshop: Formal Models of Agents, Sesimbra, Portugal 1996.
[42] A. Lomuscio, M. J. Sergot: On Multi-agent Systems Specification via Deontic Logic. ATAL 2001. International workshop No8, Seattle WA , ETATS-UNIS , vol. 2333, pp. 86-99, 2002.
[43] F. Mokhati, M. Badri, L. Badri: A Formal Framework Supporting the Specification of the Interactions between Agents. Informatica (Slovenia) 31(3). pp. 337-350. 2007.
[44] B. Chen, S. Sadaoui. A Generic Formal Framework for Multi-agent Interaction Protocols. Technical Report TR 2004-05 ISBN 0-7731-0483-6 Department of Computer Science, University of Regina, Regina SK, Canada, 2004.
[45] M. Wooldridge, N. Jennings and D. Kinny, The GAIA Methodology for agent-oriented analysis and design, Autonomous Agents and Multi-Agent Systems 3(3) (2000) 285-312.
[46] A. Mohammed, U. Furbach. Multi-agent Systems: Modeling and verification Using Hybrid Automata. In Lars Braubach, Jean-Pierre Briot, and John Thangarajah, editors, Revised and Invited Papers of the post-proceedings of 7th International Workshop on P rogramming Multi-Agent Systems (ProMAS2009), LNAI 5919, pages 49-66, Springer.
[47] Projet SPECTRE : Spécification et programmation des systèmes communicants et temps réel. Rapport d'activité INRIA 1996.
[48] A. Benzakour. Vérification formelle des systèmes parallèles, Mémoire présenté à l a Faculté des études supérieures de l'université Laval pour l'obtention du grade de Maître ès Sciences. 1997.
[49] Duboz R., D. Versmisse, G. Quesnel, A. Muzzy, E. Ramat. Specification of Dynamic Structure Discret event Multiagent Systems 2006 A gent-Directed Simulation (ADS 2006). Huntsville, AL, USA, April 2-6 2005.
[50] H. Lin, Designing Multi-Agent Systems from Logic Specifications: A Case Study, in Vijay Sugumaran (ed.), Distributed Artificial Intelligence, Agent Technology, and Collaborative Applications, IGI Global, 2008, pp. 1-27.
[51] V. Mascardi. M. Martelli and L. Sterling. Logic-Based Specification Languages for Intelligent Software Agents. Theory and Practice of Logic Programming Journal (TPLP). Volume 4 Issue 4, July 2004. publisher Cambridge University Press, pp. 429-494.
[52] A. Lomuscio, M. J. Sergot: On Multi-agent Systems Specification via Deontic Logic. ATAL 2001. International workshop No8, Seattle WA , ETATS-UNIS , 2002 vol. 2333, pp. 86-99.
[53] Bonabeau, E. Agent-based modeling: methods and techniques for simulating human systems. In *Proc. National Academy of Sciences* 99(3): 7280-7287. 2001.
[54] P. Bommel. Définition d'un cadre méthodologique pour la conception de modèles multi-agents adaptée à la gestion des ressources renouvelables. Thèse de doctorat en informatique

de l'université de Montpellier II-Sciences et Techniques du Languedoc. 2009.
[55] Ramat, E. Introduction à la modélisation et à la simulation à événements discrets. In : Modélisation et simulation multi-agents pour les Sciences de l'Homme et de la Société, Amblard F. and Phan D. (eds.), Londres, Hermes-Sciences & Lavoisier, ISBN : 2-7462-1310-9. 2006.
[56] D. T. Ndumu, H. S. Nwana, Research and development challenges for agent-based systems. IEE Proc. of Software Engineering, 144(1), 1997.
[57] M.O. Stehr, J. Meseguer, and Peter C, Olveczky. Rewriting Logic as a Unifying Framework for Petri Nets. In Unifying Petri Nets. Lecture Notes in Computer Science (Advances in Petri Nets), 2001.
[58] A. Verdejo and N. Marti Oliet. Executing E-LOTOS processes in MAUDE. In H. Ehrig, M. Grosse-Rhode, and F. Orejas, editors, INT 2000, Integration of Specification Techniques with Applications in Engineering, Extended Abstracts, pp 49-53. Technical report 2000/04, Technische Universitat Berlin, March 2000.
[59] A. Verdejo and N. Marti Oliet, Implementing CCS in MAUDE. In T. Bolognesi and D. Latella, editors, Formal Methods For Distributed System Development. FORTE/PSTV 2000 I FIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communications Protocols (FORTE XIII) and Protocol Specification, Testing and Verification (PSTV XX), Pisa, Italy, Kluwer Academic Publishers, pp 351-366, October 2000.
[60] V. Lopez, J. Alberto, N. Marti Oliet, Executing and verifying CCS in MAUDE. Technical report, pp 1-47, 99-00
[61] M.O. Stehr and C. Talcott, PLAN in MAUDE: Specifying an active network programming language. In F. Gadducci and U. Montanari, editors, Proc. 4th. Intl. Workshop on Rewriting Logic and its Applications. ENTCS, Elsevier, 2002
[62] P. Thati, S. Koushik, N. Marti Oliet, An Executable Specification of Asynchronous Pi-Calculus Semantics and May Testing in MAUDE 2.0. In 4th International Workshop on Rewriting Logic and its Applications (WRLA'02).
[63] J.-C. Fernandez, C.Jard, T.Jron, C.Viho, Using on-the-fly verification techniques for the generation of test suites, in Proceedings of Conference on Computer-Aided Verification (CAV '96), LNCS 1102, pp. 348-359, Springer, 1996.
[64] G. Bhat, R. Cleaveland, O. Grumberg, Efficient on-the-fly Model checking for CTL*, in Prooceedongs of Symposium on Logics in Computer Science, pp.388-397, IEEE, 1995.
[65] Stirling C, Walker D. Local model checking in the modal μ-calculus. Theoretical Computer Science, In: Dıaz,. J., Orejas, F. (eds.) Proceedings of the International Joint Conference on Theory and Practice of Software Development (TAPSOFT 1989), Barcelona, Spain, vol. 354, pp. 369–383. Springer, Berlin 1989.
[66] Klaus Havelund, Arne Skou, Kim G. Larsen, and Kristian Lund. Formal modeling and analysis of an audio/video protocol : An industrial case study using UPPAAL. In Proc. 18th IEEE Real-Time Systems Symposium (RTSS'97), IEEE Computer Society Press, pp 2–13, 1997.
[67] Stavros Tripakis and Sergio Yovine. Verification of the fast reservation protocol with delayed transmission using the tool KRONOS. In Proc. 4th IEEE Real-Time Technology and Applications Symposium (RTAS'98), IEEE Computer Society Press, pp 165–170, 1998.

Ammar Boucherit is an assistant teaching of computer science at the Department of Mathematics and C omputer Science of the University center of El-Oued in Algeria. He holds the Magister (master's degree) in software engineering from University of Oum El-Bouaghi. His research focuses on t hree areas: formal design and verification of computer systems, software architecture and agent based modeling (ABM).

Abdallah Khababa is currently an as sistant professor at the university of Ferhat abbas Setif, Algeria. He holds a Ph.D. in computer science from the University of Ferhat abbas Setif. His main areas of interest include cognitive science and artificial intelligence, knowledge representation with ontologies and human-machine interface and its usability.