# Rewriting Logic based Approach for the Formalization of Critical Systems based on Multi-Agent System

Ammar Boucherit
Computer Science Department,
University of Ferhat Abbas, Setif,
Algeria

Abdallah Khebaba
Computer Science Department,
University of Ferhat Abbas, Setif,
Algeria

Faiza Belala
Computer Science Department,
University of Mentouri, Constantine,
Algeria

## ABSTRACT
The agent-oriented paradigm is an emerging technology, which has significant and growing interest, particularly through its ability to be used in the modeling of all types of systems and representation of knowledge.

However, this potentiality should not hide the difficulties associated with them in the design and verification, which may cause the scientific credibility of multi-agent modeling field, especially for the case of embedded and critical systems.

In this paper, we propose a new formal approach based on rewriting logic, in which we attempt to bridge the gap between agent based system analysis and its specification. In addition, our approach includes a well-known and effective verification technique, model checking, and allows independent of the used formalism to verify an important number of properties deemed relevant on critical system based on agent paradigm.

## General Terms
Software Engineering, Agent-Based System.

## Keywords
Critical System, Model-Checking, Multi-agent systems, Maude, Rewriting logic, Specification, Testing, Verification.

## 1. INTRODUCTION
Firstly, if we simply put that, a system is an organized collection of parts (or subsystems) that are highly integrated to accomplish an overall goal. System modeling is the process, which we show how the system should be working. The use of this technique is to examine how various components work together to produce a particular outcome.

Secondly, nowadays applications (or systems) are strongly characterized by their complexity. They are usually composed by heterogeneous and distributed entities, which must cooperate and coordinate in an "intelligent" way to exchange and share knowledge, in order to solve problems which are difficult or impossible for an individual entity.

The paradigm of multi-agent systems [41, 42], which offers an original way of modeling, is considered as an appropriate method that faces the problem of modeling such kinds of applications. Therefore, multi-agent based modeling method is present in the most of sectors: telecommunications, finance, Internet, energy, health, embedded systems ... etc.

Thirdly, the potential of multi-agent systems 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.

Therefore, 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 this paper, we present an efficient formal approach based on rewriting logic formalism by using its language "Maude", and includes a well-known and effective verification technique, model checking. In fact, this approach is the extension and the improvement of our previous work [01, 02].

## 2. PRELIMINARIES
In this section, we first present some preliminaries and definitions related to the work to be presented in this paper.

### 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 [35], and the set of these agents, with these various capacities constitute *a Multi-Agents System* (MAS).

Various definitions from different disciplines have been proposed for the term multi-agent system. As given in [40], " 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 some domains require the aptitude and competence of a set of agents, in order to solve problems, which are difficult or impossible for an individual agent. In addition, agents can model complex systems, and the agent-based modeling of critical industrial applications works better than other approaches. For example, in a production factory, the behavior of a complex machine that has own internal situations, its own rhythm, different reactions in different situations, can be effectively modeled by an agent.

Finally, even if the multi-agent systems offer an original way of modeling, and their uses are very different in practice, because of its promise as a new paradigm for designing software and systems. We can resume the inherent difficulties in three points:

1) At this time, there is no evidence of a well-established engineering approach 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 [03].
3) It would be practically impossible to develop a universal "MAS Library" and design generic 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 critical systems based-agent before its implementation. These methods should not be limited to one phase, but it must cover all the process of their development, in order to prove the safety of models intended to represent the relevant functions of the system.

## 2.2 Model-Checking

Model checking is a formal verification technique [05, 06, 07], that determines whether given properties φ of a system are satisfied by a model M, where a model is defined as a formal representation of the real world [04]. We write $M \models \varphi$ as a judgment and say a model checker verifies or refutes such judgments, based on a partial or exhaustive exploration of the state space of the model. In other words, this formal verification technique analyzes the reliability, performance and checks the consistency between a property specification and a behavior model of the system. Its main objective 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 figure (Fig. 1) supports two different levels of specification: (1) a system specification level, in which the concurrent system to be analyzed is formalized; and (2) a property specification level, in which the properties to be model checked are specified. On the other hand, model checker outputs either a claim that the property is true or a counter example reporting the inconsistency. A counterexample is an execution trace of the state machine showing how the predicate is false.

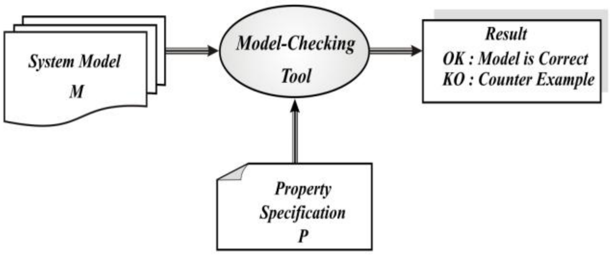


**Fig 1. Model Checking Approach**

Currently, the "on the fly" or "symbolic" model checking are the most common used. These approaches, initially introduced to overcome the problem of infinite state machines. The big advantage of the on-the-fly approach is that hopefully only a fragment of the overall state space might need to be generated and analyzed to be able to produce the correct result [36][37]. Contrary to classical methods, their effectiveness has been demonstrated, and they were used to analyze real systems of significant size [33, 34].

## 2.3 Rewriting Logic

Rewriting logic is a computational logic proposed by Meseguer [13] as a unified logic for (true) concurrency, which builds upon equational logic by extending it with "*rewrite rules*" to adapt it to changes [10], and specification of concurrent systems. In other words, 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 [12], Labeled Transition Systems [13], E-LOTOS [14], CCS [15, 16], PLAN [17], Pi-Calculus [18] … etc.

In rewriting logic, a concurrent system can be specified easily by a rewriting theory. A *rewrite theory R* is defined as a 4-tuple $R = (\Sigma, E, L, R)$ where : $(\Sigma, E)$ is an equational theory, $L$ is a set of labels, and $R$ is a set of possibly conditional labeled *rewrite rules*, $t \rightarrow t'$ that are applied modulo the equations $E$. Intuitively, the signature $(\Sigma, E)$ of a rewrite theory describes a particular structure for the states of a system, and the rewrite rules describe which elementary local transitions are possible in the distributed state by concurrent local transformations if a condition C is verified [11,13].

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

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

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

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

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

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

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

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

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

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

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**

## 2.4 Maude System

Maude [38] is a high-level language and a high-performance system supporting executable specification and declarative programming in rewriting logic. Maude is based on rewriting logic where the object systems from simple to more complex models are specified easily by the use of the theory of concurrent objects. 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 has a collection of formal tools supporting different forms of verification such as:

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

## 2.5 The Maude's LTL Model-Checker

Model-checking is as what we said previously, an automatic method for deciding if a circuit, program or a specification model, expressed as a concurrent transition system, satisfies a set of properties expressed in a temporal logic such as LTL. The Maude's LTL model checker is a very powerful model checker. It was designed with the goal of combining a very expressive and general system specification language (Maude) with an advanced on-the-fly explicit-state LTL model-checking engine. The main modules used by the Maude's LTL Model-Checker are presented in the figure (Fig. 2).

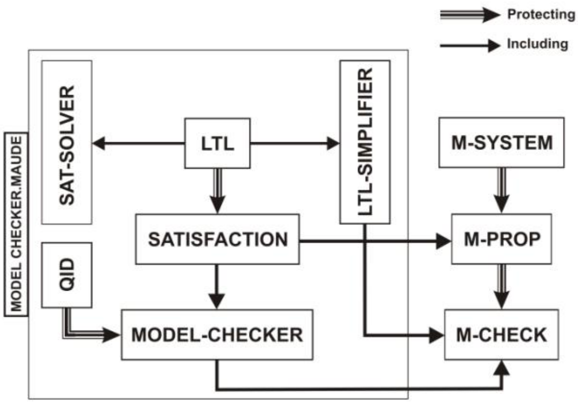


**Fig 2. The Main Modules of Maude's LTL Model-Checker**

In Order to verify such a property, the Maude's LTL model checker takes as inputs the following modules, which are defined by the user:

1. Rewrite theory specified by a Maude system module M-SYSTEM, which describing 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.
3. The initial state from which the model checker starts checking, is specified in module M-CHECK.

In addition to modules defined by user, the Maude's LTL model checker includes other modules that have well defined roles:

- 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);
- LTL SIMPLIFIER : It tries to further simplify the negative normal form of the formula $\neg\varphi$ : in the hope of generating a smaller Büchi automaton $B_{\neg\varphi}$;
- 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. CRITICAL SYSTEM FORMALIZATION

When we want to talk about the formalization of critical systems, it is strongly advised to explore the attempts of formalization of other systems that can be considered as critical systems, such as real-time systems, parallel and complex systems. In addition, because the agent-based modeling is one of the most used approaches and it works better than other approaches in the case of critical systems. We will focus in this section on formalization of multi-agent based systems.

In the last two decades, multi-agent systems have both become widely applied and increasingly complex. Therefore, a lot of approaches, languages and methods have been proposed to face the problem of developing agent-based systems [43, 56].

In this section, we will present the works that we are seeing significant in the field of specification and verification of multi-agent systems. Then, we will try to summarize the previous attempts of formalization, in order to reveal the advantages and the limitations of either kind of approach.

## 3.1 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 languages were proposed, we give briefly here four examples:

### *A. CASL Specification Language*

The Cognitive Agents Specification Language (CASL) is a framework for specifying Multi-agent systems, which allows the specifier to view agents as entities with mental states, such as knowledge, beliefs, and goals, and to define the behavior of the agents in terms of their mental states [44]. It combines two powerful components. The first one is a declarative action theory, which allows the specifier to describe the effects of actions on the world and the mental states of agents. The second component is a rich programming/process language with constructs for concurrency and non-determinism to facilitate the specification and verification of multi-agent systems.

### *B. AUML*

The best-known initiative to extend UML with facilities for describing agents called AUML. It starts from the idea that multi-agent systems are often characterized as extensions of object-oriented systems. In other words, if the unified modeling language (Unified Modeling Language) is an attempt to unify the different paradigms of analysis and design object oriented software and provide a unique notation for modeling object-oriented systems, the AUML was proposed to adapt the UML notation to describe the agent-oriented modeling [45,46].

### *C. The Agent Modeling Language: AML*

The Agent Modeling Language (AML) is a semi-formal visual modeling language, specified as an extension to UML 2.0. AML is designed to capture the aspects of multi-agent systems. The ultimate objective for AML is to provide a means for software engineers to incorporate aspects of multi-agent system engineering into their analysis and design processes. In other words, AML is designed to support business modeling, requirements specification analysis, and design of software systems based on software agent concepts and principles [55, 58].

### *D. SLAB Language*

In his paper [52], the author was presented a powerful formal specification language (SLAB) for multi-agent systems. The (SLAB) language integrates a number of novel language facilities that support the development of agent-based systems. In order to show that these facilities are powerful and useful for the formal specification of agents in various models and theories, the author specified example systems of agent-based systems in SLAB.

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: [19, 20]:

- **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, are examples [19, 48].

- **Logic 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 [47, 50, 54].

## 3.2 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 system 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.

Finally, we can present in the figure (Fig.3) non-exhaustive list for the attempts were found in the literature for the formalization of multi-agent systems and the used formalisms.

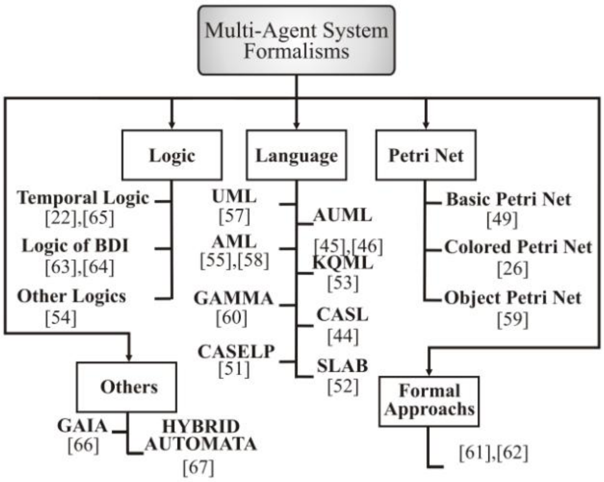


**Fig 3. Formalisms used for the Formalization of MAS**

## 3.3 Synthesis

First, we have to note that in our opinion, the two specification approaches are complementary, and their combination can be very interesting, as it is important to adopt the most appropriate formalism for the representation of the system. We justify this idea by:

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. In addition, the combined analysis of static and dynamic aspects of a system is also necessary for detecting hot spots in 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 not belong to the same approach in the mentioned classification can be found together in an other approach 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, because we are interested by the agent based design, we can also find in the literature, several attempts at 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 :[21, 22, 23, 24, 25, 27, 28, 49], etc.

In the case of multi-agent systems, the specification is to develop an abstract model of the real system. The interest of a model is initially to be more explicit, simpler and easier to manipulate than the reality it is supposed to represent. Moreover, the specification of multi-agent systems must be based on a powerful operational and unambiguous formalization. Nevertheless, in the view of the absence of a consensus on the most suitable formalism for specifying multi-agent systems, we have to note here our agree with the ideas of [29, 30], that there is no perfect model, and we are wrong if we think that the goal is to offer the most complete model and the most "beautiful", because the reality is always more complex than we imagine.

Therefore, we must build the system model with the most suited formalism to check the properties in question, and not to limit to the use of a single formalism. Because, as noted in [31], if we take the example of the paradigm of multi-agent systems; the specification of system structure can be performed using UML, while the dynamics of the agents may be specified using Petri nets or inference rules.

Finally, it is crucial to search for formalisms that allow full description of the multi-agent based system and the consistent expression of its different aspects: structure, behavior, control ... etc. In addition, a set of relevant properties of the system must be verifiable with the proposed formalisms by using effective tools. Because, as noted in [32]: "any sufficiently complex system has consequences that exceed its capabilities of proof" Therefore, the use of such a method is needed from the initial specification to implementation.

## 4. FORMALIZATION APPROACH FOR CRITICAL SYSTEMS BASED ON MULTI-AGENT SYSTEMS

In system design, the process of verification and validation can be too complex, especially when it depends to ensuring that the system has no failures (unexpected behavior) and that it meets its specifications correctly. Indeed, in the case of designing critical systems, the steps of formal specification and verification are essential to avoid any type of error and validate systems before their implementation. The specification phase is intended to clearly express all the expected features of the system, while the integration of the verification phase in the design process can detect the error once it appears, and it allows to avoid repeating all the verification process by reusing intermediate results.

### 4.1 Global Description

In our approach, which is based on the use of formal and automatic techniques, we start from a specification written in rewriting logic of the proposed model for the system, and a specification of the expected properties, in order to determine whether the system model satisfies the properties in all its possible executions. We present in the following figure Fig.4, the steps of the proposed approach for the verification of relevant properties of critical systems.

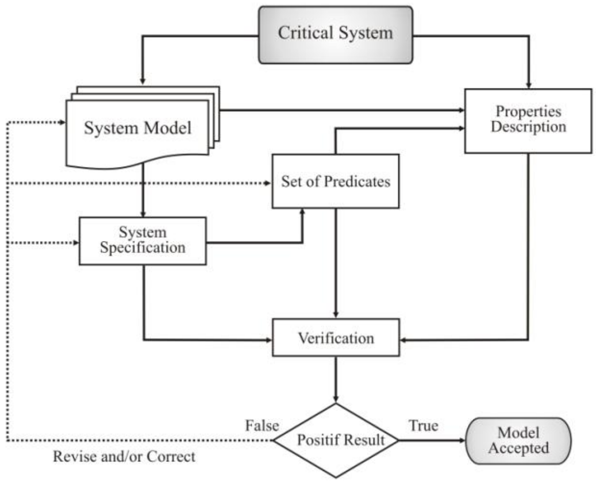


**Fig 4. Global Description of the Formalization Approach**

### 4.2 Detailed Description

Our approach for the formalization of multi-agent based systems can be summarized into three essential steps:

***Step 01: (System Specification)***

The purpose of this step is to describe the full specification and to express all the expected features of the system. We note here that in the case of multi-agent systems, the first step of specification is to develop a model clearly and unambiguously. Using one of the most used formalisms such as UML [57], Petri nets, labeled transition systems … etc

In our approach, we will not be limited to use only one specification approach or a single formalism, but according to the aspect or the property to check, we will choose the most adapted formalism to the case study. In other words, it is very judicious to use several formalisms for the same system to take advantages of each formalism and verify a large number of system properties. [29, 30].

This stage ends with a description of each model in rewriting logic, which is logic of change and a unifying semantic framework. Taking advantage of its expressiveness and powerful tools built into its system Maude. A description of this step is illustrated in the following figure Fig 5.

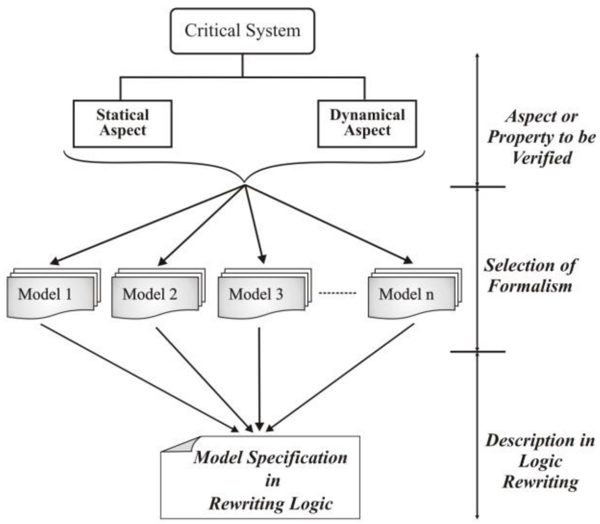


**Fig 5. Description of the Specification Step**

***Step 02: (Properties Specification)***

If the aim of the first step, 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 must 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. We always refer to the proposed model and its specification of the first step. Then, the set of properties to be checked must be also expressed by using linear temporal logic.

***Step 03: (Verification)***

Finally, a 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.6, are applied to perform this step:

**1- 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.

**2- Empirical Test :**
This time, we use another Maude's tool, which is: *Search*. Its use is based on situations and empirical cases offered by experts in the field; in order to confirm 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.

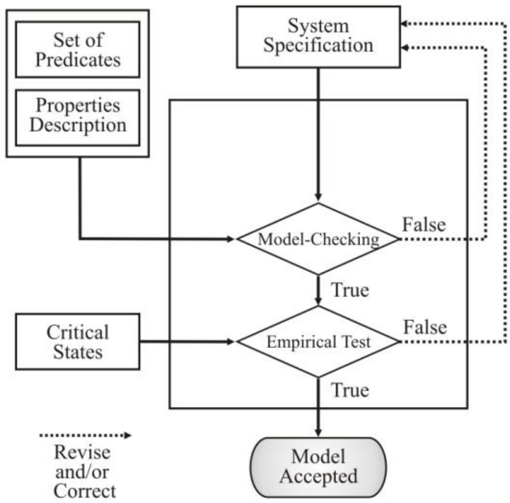


**Fig 6. Description of the Verification Step**

## 5. CONCLUSION AND FUTURE WORK

Research in the field of multi-agent systems (MAS) is becoming increasingly important, particularly through its ability to model all types of systems. However, the potential of multi-agent systems (MAS) should not hide the difficulties associated with them in the design and verification, especially for the case of critical systems. Formal methods have been proposed as mathematical techniques to help the designer to solve this problem. Nevertheless, each of these methods is used to solve a specific class of problems, depending to the type of formalisms used.

In this paper, we have extended our previous approach [1, 2], in order to provide a more comprehensive approach based on rewriting logic for the specification and verification of critical systems based agent, including model checking technique and the technique of empirical test. Our approach allows to verify a large number of properties of a critical system regardless of the formalism used for the specification. In other words, our approach tends to provide a full specification for critical systems based MAS, leaving the choice to the user to adopt the most appropriate formalism for the representation of models and the expression of properties.

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. Third, 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 directly 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 axis of 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 the DSSE member's and all member of LIRE Laboratory for their precious remarks, helps and their contribution to preparing this work.

## 7. REFERENCES

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

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

[3] 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.

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

[5] 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, June 1986, pp 332–344.

[6] 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, 1985, pp 97–107.

[7] 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, April 1986, pp 244–263.

[8] N. Marti-Oliet, J. Meseguer, Rewriting Logic as a Logical and Semantic Framework, Electronic Notes in Theoretical Computer Science, Vol 4, no1, 1996, pp1-36.

[9] N. Marti-Oliet, J. Meseguer, Rewriting Logic as a Logical and Semantic Framework, Technical Report SRI-CSL-93-05, Menlo Park, CA 94025, and Center for the study of language and Information Stanford University, Stanford, CA 94305, 1993.

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

[11] J. Meseguer. Rewriting Logic Revisited, Slides of tutorial presented at WRLA 2002, Pisa, Italy, September 2002.

[12] M.O. Stehr, José Meseguer, and Peter C. Ölveczky. Rewriting Logic as a Unifying Framework for Petri Nets.

In Unifying Petri Nets. Lecture Notes in Computer Science (Advances in Petri Nets). 2001.

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

[14] A. Verdejo and N. Mart-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.

[15] A. Verdejo and N. Mart -Oliet. Implementing CCS in MAUDE. In T. Bolognesi and D. Latella, editors, Formal Methods For Distributed System Development. FORTE/PSTV 2000 IFIP 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) October 2000, Pisa, Italy, Kluwer Academic Publishers, 2000, pp 351-366.

[16] V. López, J. Alberto, N.Martí Oliet, Executing and verifying CCS in MAUDE. Technical report, 99-00. pp 1-47.

[17] 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.

[18] 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).

[19] Projet SPECTRE : Spécification et programmation des systèmes communicants et temps réel. Rapport d'activité INRIA 1996.

[20] A. Benzakour. Vérification formelle des systèmes parallèles, Mémoire présenté à la Faculté des études supérieures de l'université Laval pour l'obtention du grade de Maître ès Sciences. 1997.

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

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

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

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

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

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

[28] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. Logic Journal of the IGPL, 11(2):133-157, March 2003.

[29] 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.

[30] 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.

[31] G. Quesnel. Approche formelle et opérationnelle de la multi-modélisation et de la simulation des systèmes complexes. Thèse de doctorat en informatique à l'école doctorale de l'université du Littoral - Côte d'Opale. 2006.

[32] H. ZWIRN Les limites de la connaissance, Paris, Odile Jacob, 2000.

[33] K. Havelund, A. Skou, G. Larsen, K. 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.

[34] S. Tripakis, S. 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.

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

[36] 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.

[37] 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.

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

[39] S. Eker, J. Meseguer, A. Sridharanarayanan. The Maude LTL Model Checker. Electronic Notes in Theoretical Computer Science, Volume 71. From Proceedings of the 4th International Workshop on Rewriting Logic and Its Applications (WRLA 2002). Edited by Fabio Gaducci and Ugo Montanari. Elsevier, Amsterdam. September, 2002.

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

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

[42] D. T. Ndumu and H. S. Nwana. Research and development challenges for agent-based systems. IEE Proc. of Software Engineering, 144(1), 1997.

[43] M. Dastani, K. V. Hindriks, J.C. Meyer. Specification Language and Verification Environment. 1st Edition, 2010, XVII, 405 p. 100 illus., Hardcover. Publisher: Springer. ISBN: 978-1-4419-6983-5.

[44] S. Shapiro, Y. Lespérance, and H.J. Levesque. The cognitive agents specification language and verification environment for multiagent systems, in Proc. AAMAS, 2002, pp.19-26.

[45] 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.

[46] 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.

[47] 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.

[48] Duboz R., D. Versmisse, G. Quesnel, A. Muzzy, E. Ramat. Specification of Dynamic Structure Discret event Multiagent Systems 2006 Agent-Directed Simulation (ADS 2006). Huntsville, AL, USA, April 2-6 2005.

[49] 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.

[50] 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.

[51] M. Martelli, V. Mascardi, Floriano Zini. Specification and Simulation of Multi-Agent Systems in CaseLP. APPIA-GULP-PRODE'1999. pp.13-28.

[52] H. Zhu, SLABS: A Formal Specification Language for Agent-Based Systems, International Journal of Software Engineering and Knowledge Engineering, Vol. 11. No. 5, pp529~558.

[53] Finin, T., Labrou, Y.: KQML as an agent communication language. In J.M. Bradshaw (ed.), Software Agents, MIT Press, Cambridge, MA, (1997), 291-316.

[54] 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.

[55] R. Cervenka, I. Trencanský, M. Calisti: Modeling Social Aspects of Multi-Agent Systems: The AML Approach. AOSE 2005. pp. 28-39.

[56] L.S. Sterling, K.Taveter, The Art of Agent-Oriented Modeling. The MIT Press 2009.

[57] 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.

[58] I. Trencansky and R. Cervenka, Agent Modeling Language (AML): A comprehensive approach to modeling MAS, *Informatica* 29(4) 2005 391-400.

[59] 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) 57-61.

[60] H. Lin and C. Yang, C. Spécification de systèmes multi-agent dans le langage Gamma. Proceedings of the IEEE 19th Annual Canadian Conference on Electrical and Computer Engineering (CCECE05). Ottawa, Ontario, Canada. Du 7 au 10 mai 2006. Numéro de publication du CNRC : NRC 48476.

[61] 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.

[62] 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.

[63] 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.

[64] M. Wooldridge, 1996. "A logic for BDI planning agents" In Pierre-Yves Schobbens, editor, *Working Notes of 2nd ModelAge Workshop: Formal Models of Agents*, Sesimbra, Portugal

[65] M. Fisher, 1996. An introduction to executable temporal logics. *Knowledge Engineering Review*, 11(1). pp. 43–56.

[66] 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.

[67] 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 Programming Multi-Agent Systems (ProMAS2009), LNAI 5919, pages 49-66, Springer.