A Tool for Visualizing ARTIS Agents Specification

Toufik MARIR
Oum El Bouaghi University
Oum El Bouaghi
Algeria

Farid MOKHATI
Oum El Bouaghi University
Oum El Bouaghi
Algeria

Zina MECIBAH
Oum El Bouaghi University
Oum El Bouaghi
Algeria

ABSTRACT
Real-time agent based systems are characterized by their complexity in which several skills are required in all development process stages. Specially, the communication between the different development groups is a vital task. The graphical specifications play an interesting role in understanding system aspects. In this paper we propose the transformation of XML based ARTIS agent specifications to graphical representations. Using our approach we can visualize the ARTIS agent specifications in informal graphical representation designed for all kinds of users or in based object UML diagram designed for specialist users. Moreover, a tool is developed to support the visualization process.

General Terms
System Understanding, Graphical Visualization.

Keywords
Real Time Agents, ARTIS Model, MAS Understanding, Object UML Diagram, XML.

1. INTRODUCTION
Real-time agents represent an ideal solution for developing complex systems in which real time aspects play a primordial role. Providing important perspectives [01], real-time agents are used to develop a wide range of applications such as avionics, adaptive control, robotic and computer vision [02]. However, the applied methodologies for developing real-time agents still immature [03]. Even though formal methods are suggested to develop critical systems, the use of formal methods requires more time and effort [04]. For this purpose, graphical specifications can be shown as a complement to formal methods. In fact, graphical specifications provide, among others advantages, an ideal basis to assist the communication activity among the stakeholders in a better and faster way [05, 06]. In this paper we aim to transform XML specification of ARTIS agents to graphical representations that makes its understanding and its exploitation easier.

In order to make agent respect real-time constraints, several models are proposed like: RT1 [07], ANYMAS [08], ARTIS [09] and SIMBA [10]. In order to visualize the structural aspects of given systems, the proposed approach should target known model of real-time agent. Our approach is based upon ARTIS architecture which is one of the well-known real-time agent models. In fact, ARTIS model is composed of two levels of agents: an ARTIS agent and a set of internal agents (called in-agent) [09]. Among other features, ARTIS agent guarantees all the critical time requirements of reflex level by means of off-line scheduling. Moreover, this model has strong methodologies basis consisting in the: RT-MESSAGE [11] methodology which is a methodology to develop based ARTIS real-time multi-agent systems, InSIDE tool which is a tool to designing and debugging ARTIS agent [12] and several attempts to formalize ARTIS agent [13, 14].

Despite all these features, the existence of several internal agents within the ARTIS agent that are made in turns by several knowledge sources that make this agent model too complex. Hence, the graphical specification of ARTIS based systems is more than necessary. In our approach we transform ARTIS based systems written in XML language to visual graphical specifications. XML language is a standard to interoperability inter applications allowing exchange and integration data [15]. The target visual representation is either informal graphical representation for non-specialist users or an object diagram of UML for more specialist users.

The remainder of this paper is organized as follow: In section 2, we give a brief overview of related works. Section 3 presents the ARTIS agent model followed by the description of XML language (in section 4). In section 5, we explain the visualization process. The developed tool is presented in section 6. Finally, we give a conclusion and some future works in section 7.

2. RELATED WORK
Software products become more and more complex and the time devoted to their development process increases significantly with the software complexity. Making the representation of software easy can help in the development of this latter. In fact, the appropriate graphical representation provides a basis for understanding the software in better and faster way. By appropriate graphical representation we mean a representation that takes into account the perception and cognitive factors of human. Several tools are proposed to visualize different software products at different levels (requirements, design and code).

In a recent study, Caserta and Zendra [06] classified the software's aspects which can be tackled by the visualization techniques in three kinds: static aspects, dynamic aspects and evolution aspects of software. The authors give a survey on the visualization techniques of the static aspects of the software and its evolution. The remarkable lack of studies that address the agent based software in this survey is justified, in our opinion, by the rarity of works which attempt the visualization of multi-agent systems.

The multi-agent paradigm is a powerful one to develop complex systems. Hence, the visualization of the system's behavior in its macroscopic level can help in understanding and defining the system under development. Abdelaziz and al [16] used Use Case Maps (UCMs) to visualize a high level view of multi-agent medical diagnosis system. UCMs are a high-level visual representation of activities that ignore the implementation details.

Adopting the based-model architecture to develop multi-agent systems, Anna Perini and Angelo Susi [05] have developed a tool for agent-oriented visual modeling. Using based-model approach, developers need tools for specifying both structural and dynamic properties; also they need tools for transforming

the specification through development steps. Therefore, the authors provide a complete environment which allows: the informal specification in *Tropos* [17] methodology, the automatic transformation of the informal specification into formal specification, the verification of the formal specification using model checking technique and finally the results of the model checker execution can be visualized in *Tropos* notation.

According to the classification given by Caserta and Zendra [06], our work is in line with the visualization of the *static aspects* of real-time multi-agent systems. In fact, the visualization of the formal specifications of real-time multi-agent systems is, in our knowledge, almost inexistent.

Ignoring real-time aspect of ARTIS agent, we can compare our work with previous works of the formalization of ordinary agent. Thus, the visualization of ARTIS agent shares the same objective with all previous studies. In this paper, we prefer to begin with visualizing ARTIS agents' static aspects. Comparing our approach with the one proposed by Tawfig Abdelaziz and al [16], our approach is more generic. They chose a specific application domain (medical diagnosis system) and a specific notation (Use Case Maps) which can limit the use of their approach. In contrast, our study can be applied in any based ARTIS application and by any kind of users (from non-specialist to more specialist users).

Conversely to the tool presented in [05] which transform the informal specification to formal one and vice versa, our tool allows only the visualization of formal specification of ARTIS models. This restriction is justified by the existence of several studies for the use of formal techniques in ARTIS agent context [13, 14, 18]. Moreover, the tool presented in [05] cannot support the real-time aspect of real-time agent. In fact, our study is motivated by the lack of methodologies and tools addressed the real-time agent development.

3. ARTIS AGENT

ARTIS agent is an extension of blackboard architecture designed to operate in hard real-time environment [09]. It has two levels of agents: ARTIS agent (AA) and set of internal agents (called in-agents). As it is presented in figure 01, ARTIS model consists of: set of sensors and effectors to interact with the environment, set of internal agents to model the ARTIS behavior, set of beliefs organized in based frame blackboard and control module that is responsible of real-time execution of in-agents.

Two kinds of in-agents can exist belong ARTIS agent: critical and not-critical in-agents. The critical in-agents consist of reflex level (that provides a bounded time answer with minimum quality) and cognitive level (for providing more quality answer). On the other hand the not-critical in-agents have only cognitive level.

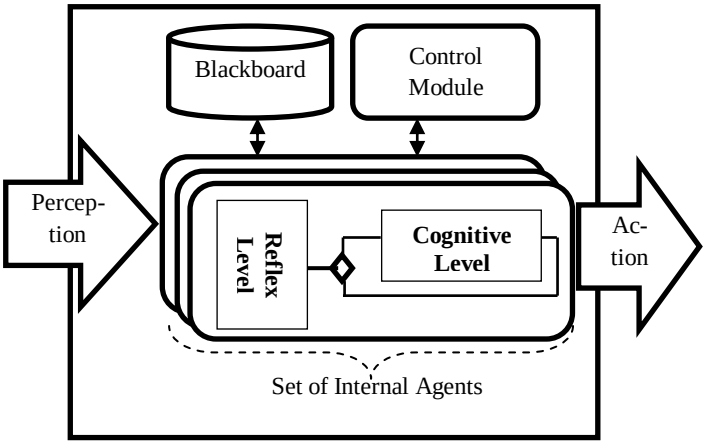

Fig 1: ARTIS agent architecture [09].

An internal agent is implemented using knowledge sources as main stones [12]. In fact, the perception and cognitive layers of in-agent is composed of several ordered knowledge sources list (called multi-level knowledge source or MKS) used to implement the anytime algorithm or multiple methods. Each MKS provides several solutions to the same problem with different levels of quality. The action layer is composed of set of knowledge sources.

The choice of ARTIS architecture as a basis of our work is motivated by many reasons:

1. It guarantees all hard real-time constraints by means of off-line scheduling,
2. It is organized on several levels of abstraction (from simple knowledge source to complex ARTIS agent),
3. It uses both well-known techniques of real-time artificial intelligence (anytime algorithm and multiple methods),
4. It is a model in full evolution taking into account its several extensions like SIMBA [10] and RT-MESSAGE [11].

Several works attempt to profit from the recent advances of formal methods in the development of ARTIS based application [13, 14, 18]. However, the complexity of formal methods is still a real obstacle for non-expert users. For this reason, we think that a visual representation of ARTIS agent can be a complement to formal methods in order to benefit from both advantages formal and graphical notations.

4. XML LANGUAGE

XML language (eXtensible Markup Language) is based markup language designed mainly to allow the change of information [19]. In fact, the XML documents consist, simply, of sequence of nested tagged elements. In contrast with HTML language, XML can be considered as self defining tags. In this way, the users can define an element using start tag (e.g ***<Example>***) and end tag (e.g ***</Example>***). The structure of the document and the constraints of its elements can be defined in the form of DTD (Document-Type Definition) or XML sche-

ma. We characterized XML document as valid if this latter respect its DTD document [20].

Considering XML as a standard to exchange the information, this language becomes quite popular. In fact, the XML is considered now as a meta-language in which we can define and represent other languages [21]. Moreover, the XML language is easy, extendable and portable language.

In the multi-agent context, XML language is used for several purposes. First of all, we can cite the XABSL language [22] (Extensible Agent Behavior Specification Language) which is a based XML language for designing agent behavior. Korzyk [23] proposed the use of XML as a secure intelligent agent communication language. In their e-learning system, Garro and Palpoli proposed a based XML multi-agent system [24].

In this work, we chose the use of XML as a language for the specification of ARTIS agent. Thus, we can integrate our tool with other based XML tools. Moreover, the specification of ARTIS on XML language can be considered as unified framework in which we can represent all previous formalization of ARTIS agent.

5. VISUALIZATION PROCESS

This section is devoted to the visualization process we propose in which the XML specifications of ARTIS based systems are transformed in graphical representation. However, we should begin with the presentation of XML specification of ARTIS agent.

5.1 XML based specification of ARTIS agent

As we noted above, ARTIS agent can be specified in several methods and language like RT-MESSAGE [11], RT-Maude [14] and extension of RTCTL [13]. Considered as meta-language, XML can represent all these languages. Moreover, the XML language allows the exchange of information between different applications. For this reason we chose to specify based ARTIS agent systems in XML language.

First of all, we translate the ARTIS agent model in a DTD code (for Document Type Definition) to ensure the conformity of the specifications to the structure of ARTIS and its constraints. For paper size limitation, we present in figure 2 only a portion of this DTD model. Hence, an ARTIS agent is composed of a blackboard and a list of internal agents (Line [1]) and it has three attributes: its Identifier (Line [2]), its Timer (Line [3]) and the priority of the current activated in-agent (Line [4]). The blackboard can be either empty or includes one or more frames (Line [5]). The frame is defined by several attributes: its Identifier (Line [6]), its type which can be Timed or General (Line [7]), its apparition date (Line [8]) and its validating duration (Line [9]). The in-agent list can be either empty or includes one or more in-agent (Line [10]).

```
<!ELEMENT ARTIS_AGENT (InAgent_List, Black-
      Board)>                <!--01-->

<!ATTLIST ARTIS_AGENT ARTIS_id ID #REQUIRED >
      <!--02-->

<!ATTLIST ARTIS_AGENT Timer_Is CDATA #RE-
      QUIRED >                <!--03-->

<!ATTLIST ARTIS_AGENT Current_Priority CDATA
#REQUIRED >                    <!--04-->

<!ELEMENT BlackBoard (EmptyBlackBoard |
      Frame+ ) >               <!--05-->

<!ATTLIST Frame Frame_id ID #REQUIRED >
      <!--06-->

<!ATTLIST Frame Frame_Type (Timed | General )
#REQUIRED >                    <!--07-->

<!ATTLIST Frame Apparition_Date_Is CDATA #RE-
      QUIRED >                <!--08-->

<!ATTLIST Frame Validiting_Time_Is CDATA #IM-
      PLIED >                 <!--09-->

<!ELEMENT InAgent_List (EmptyInAgent | In-
      Agent+ ) >              <!--10-->
```

Fig 2: Portion of ARTIS structure in DTD code

Respecting the above DTD, we can specify based ARTIS systems in XML language as is presented in figure 3. In fact, this portion of XML code represents the specification of an ARTIS agent (called "ARTIS_Agent") which is composed of two internal agents (called "In_Agent1" and "In_Agent2") and its blackboard contains only one frame (called "Frame1"). For paper size limitation, the in-agents specifications cannot be presented in this figure.

```
     <!----ARTIS Agent Specification---->

<ARTIS_AGENT ARTIS_id="ARTIS_Agent" Ti-
mer_Is="15" Current_Priority="0">

   <InAgent_List>

         <!--In-Agent 01 Specification-->

        <InAgent InAgent_id = "In_Agent1" In-
        Agent_Kind = "Critical" Priority_Is =
        "0" Current_State = "Activated" Dead-
        line_Is = "100" Period_Is = "150" Ti-
        mer_Is = "46">

<!-- The remainder structure of In_Agent1-->

        </InAgent>

         <!--In-Agent 02 Specification-->

        <InAgent InAgent_id = "InAgent_2" In-
        Agent_Kind = "Critical" Priority_Is =
        "1" Current_State = "Wait" Deadline_Is
        = "100" Period_Is = "150" Timer_Is =
        "20">

<!-- The remainder structure of In_Agent2-->

        </InAgent>

   </InAgent_List>

   <BlackBoard>

         <!-- Frame 01 Specification -->

        <Frame Frame_id = "Frame1" Frame_Type
        = "Timed" Apparition_Date_Is = "5" Va-
        lididting_Time_Is = "10"/>

   </BlackBoard>

</ARTIS_AGENT>
```

Fig 3: Example of ARTIS specification in XML.

5.2 Visualization of ARTIS specification

The visualization process passes through three steps (Figure 4): the control of validity step, the pre-processing step and the visualization step. The first step consists in the verification of the compliance of the XML documents with the structure of ARTIS agent defined in DTD document. For this purpose, we use one of XML editors like OXYGEN to verify the validity of the XML documents. Naturally, any attempts to transform an invalid XML documents can generate catastrophic errors.

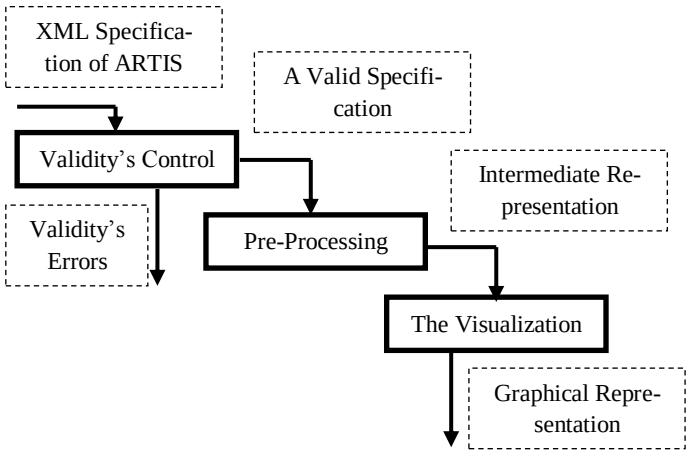

Fig 4: Visualization process of ARTIS specification.

The second step of the visualization process called pre-processing step. It consists in the generation of an intermediate representation of the specified system. We chose the oriented object paradigm to represent the specified system because of the conceptual similarity between agents and objects [25]. Thus, an ARTIS agent can be represented as an object of the class *ARTIS* with the following attributes: *identifier*, *timer*, *list of internal agents* and a *blackboard* represented as *list of frames*. An internal agent can be considered as an object of the *in-agent* class with, mainly, the following attributes: *identifier*, *list of its abilities* (**perception**, **cognitive** and **action abilities**), *list of its scheduled knowledge sources*, *its state* (**activated, suspended** or **waiting**), *the current execution knowledge source and its timer*. In the same way, knowledge sources can be derived from a specific class which contains the following attributes: *identifier*, *the execution time* and *the remainder execution time*. Moreover, the blackboard of ARTIS agent is a list of frames in which each one is an object. The main components of the intermediate representation are presented in the following meta-model (figure 5).

The last step of the visualization process is about the generation of the graphical representation of the specified system. In fact, the visualization process can generate two kinds of graphical representation: Informal representation and UML Object diagram-based representation.

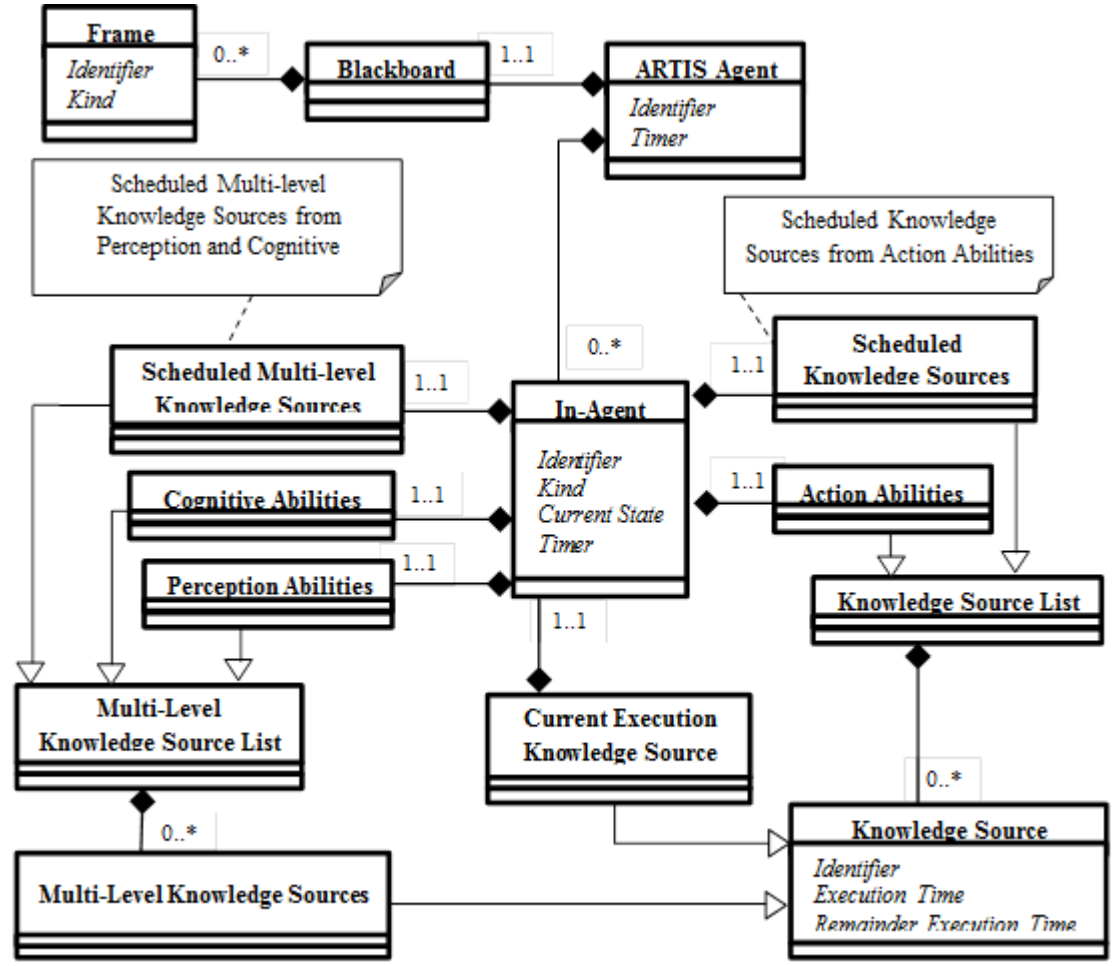

Fig 5: The intermediate representation meta-model.

5.2.1 The Informal graphical representation

By informal representation we mean a representation that is not based upon known notation and languages of software engineering. In fact, it is common to visualize the software products using metaphors and non-technical notation [06]. Adopting this approach allow to non-specialist users to understand the specified systems.

The real challenge in the visualization methods is reducing software complexity by taking into account the human perception and cognitive factors. Composed of several internal elements, ARTIS agent is complex software. In order to manage this complexity we choose to apply the Shneiderman principal (*"Overview, Zoom-in and details on demand"* [26]) in exploiting the hierarchical organization of ARTIS model. Hence, the system generates an overview of the specified system in the first time that contains the most important elements (Figure 06). Then, users can change the granularity level to focus through only one internal agent. In the same manner, the users can focus on the internal structure of an in-agent.

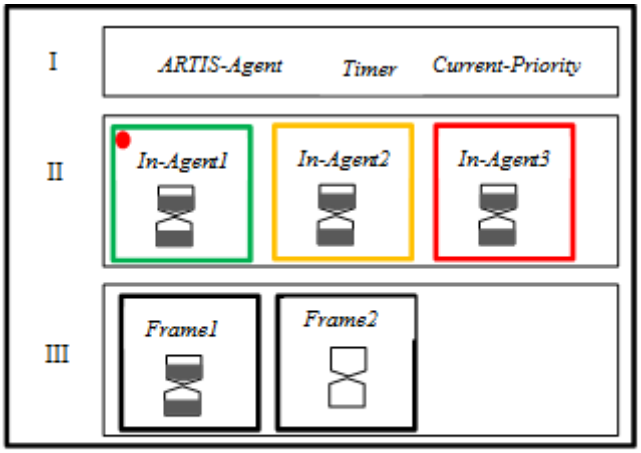

Fig 6: The top level granularity of an ARTIS Agent in informal representation.

As presented in Figure 06, the top level granularity presentation of an ARTIS agent is composed of its foremost elements:

I. The ARTIS agent's essential information (its identifier, its current priority and the value of its timer),

II. The internal agents which exist within the ARTIS agent,
III. The list of frames which exist in the blackboard structure.

In our approach we are trying to visualize the ARTIS agent in *simple* and *useful* graphical representation. A simple representation can be obtained by exploiting the known metaphors. By useful representation we mean the representation that includes the most of information. Hence, the top level graphical representation includes the following information:

- Each of **in-agent** or **frame** is represented by a square in which we found its identifier (Figure 06, for example, show three in-agents and two frames),
- The **current state** of given in-agent is represented by expressive colors of the outline of the in-agent representation. Known that the in-agent can be in one of the three states (Activated, Suspended and Wait); we choose the green color to represent activated in-agent, the red color to represent suspended in-agent and orange color to represent the wait in-agent (In figure 06, for example, the in-agent ***In-Agent1*** is in ***Activated*** state),
- The **critical nature** of given in-agent is represented by a red point in its top left corner (In figure 06, for example, the in-agent ***In-Agent1*** is critical). The lack of this point means **not critical** in-agent (for example ***In-Agent2*** and ***In-Agent3*** in figure 06),
- The **in-agent's timer** represents the remainder time in the current state. Hence, the timer of an activated or suspended in-agent is the remainder time compared by the deadline value. In contrast, the timer of a wait in-agent is the remainder time compared by the period value. For this reason, we choose to represent the timer of each in-agent by *hourglass* because it represents the remainder and the used time in the current state.
- The **frame's nature** is represented by *hourglass*. In fact, given frame can be from timed nature (with bounded validating time) or from general nature (without bounded validating time). Hence, the hourglass represents the remainder validating time for the indicate frame (like the frame ***Frame1*** in the figure 06). The void hourglass represents a general frame (like the frame ***Frame2*** in the figure 06).

We can change the granularity level to focus on the structure of an in-agent in order to visualize its perception, cognition and action abilities. Moreover, we can focus on one knowledge source from these abilities to show its internal structure.

5.2.2 The UML object diagram based representation

Remembering that we translate the specification in an intermediate representation that consists of a set of objects, this step consists in the visualization of the specification using UML object diagram. Allowing the modeling of the systems structure at a specific time, the UML object diagram is fully conform to our need in the visualization of ARTIS based on specification structure. Moreover, object diagram is part of Unified Modeling Language which is one of well-known modeling language in software engineering. Hence, the visualization of ARTIS agent in object diagram can provide both advantages: the popularity of UML and its technical precision notation.

Figure 07 shows a representation of a simple based ARTIS system using object diagram. This system is composed of a blackboard and two in-agents. Like the informal representation, we apply here the above Shneiderman principal [26]. Hence we can focus through blackboard or one in-agent to show its structure. In fact, this principal allows simplifying the eventual complex representations.

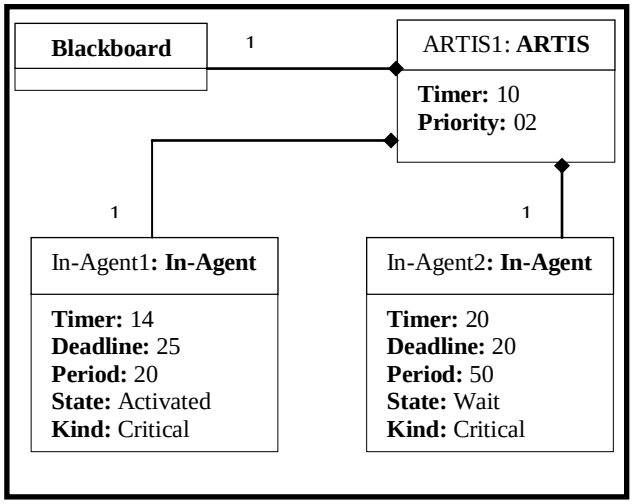

Fig 7: The top level granularity of an ARTIS Agent structure in object diagram.

6. TOOL SUPPORTING OUR APPROACH

We developed, along this work, a tool supporting the proposed visualization process. This tool allows to users to introduce and update XML specification of ARTIS based systems and the visualization of this latter.

In the first step, the users should create an XML specification or open existing specification (Figure 8). In our case study we chose to visualize the specification of the mobile robot inspired from [12]. The mobile robot is an ARTIS agent with five internal agents. The figure 8 shows a portion of the specification in which the information of the ARTIS agent (identifier is *"Mobil_Robot"*, timer is *"50"* and currant priority is *"0"*) and the information of the first in-agent (identifier is *"Avoid_Obstacle"*, timer is *"19"*, deadline is *"20*", period is *"100*", priority is "*0*", kind is "*Critical"* and state is "*Activated"*) appeared.

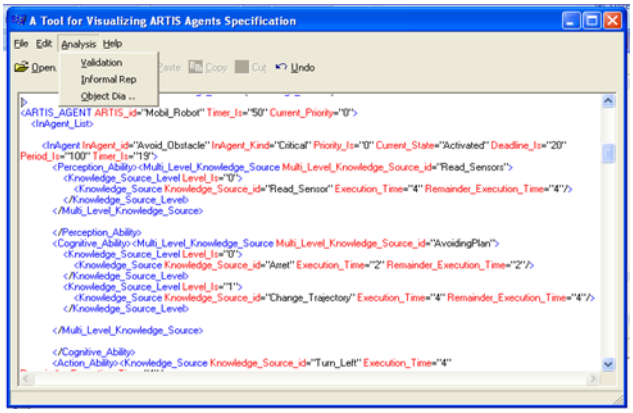

Fig 8: A portion from the case study specification.

Using our tool, we can call an XML editor (like OXYGEN tool) to validate the XML specification according to DTD (Figure 9). This step is more important in order to generate a valid graphical representation of the specified system.

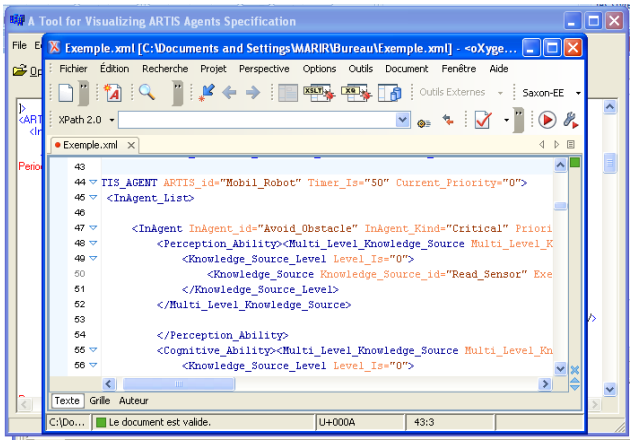

Fig 9: The validation of the case study specification.

The last step consists in the generation of the graphical representation. In the first case (Figure 10) we show a representation conform to the figure 6 in which appeared the mobile robot ARTIS agent with its five critical in-agents and three frames. As presented in the specification (figure 8) the *Avoid_Obstacle* in-agent is in activated state and its hourglass is almost full because its timer is almost equal to deadline.

In the second case we can show an UML object diagram for the same specified system (figure 11). In this object diagram we show a mobile robot object composed of a blackboard and five in-agents.

Naturally, we can focus through the composed elements in the specification (like in-agent or blackboard) by a simple click to visualize more details. However, the paper size limitation prevents the representation of more figures.

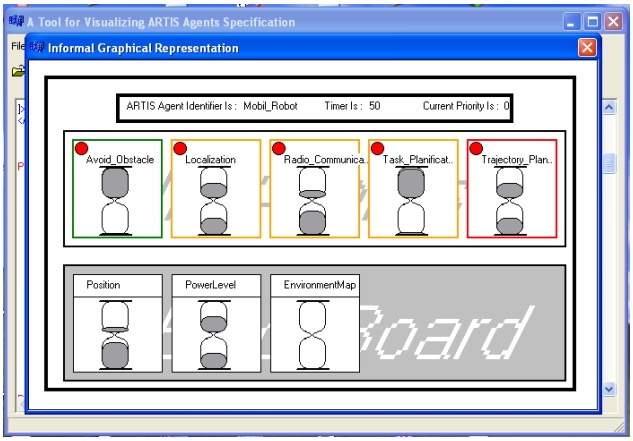

Fig 10: The top level granularity informal representation of the case study specification.

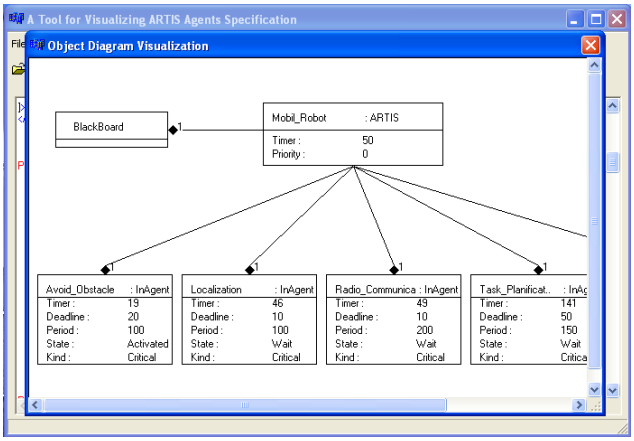

Fig 11: The top level granularity based upon object diagram representation of the case study specification.

7. CONCLUSION AND FUTURE WORKS

The visualization of formal specification play important role in the understanding of the specified systems in which formal methods are consuming time and effort. In this work we presented the visualization of ARTIS based specification. This latter is specified in XML language. Using our tool we can generate informal graphical representation based upon known metaphors for non-technical stakeholders or generate UML object diagram based representation for software engineering specialists.

Our work supports only the visualization of static aspect of the specification. We will extend this work to support both dynamic and evolutionary aspect of systems specifications. Also, we will extend this tool to support the visualization of social aspect of real-time multi-agent systems.

8. REFERENCES

[1] L. C. DiPippo , V. Fay-Wolf, L. Nair, E. Hodays and O. Uvarov, "A Real-Time Multi Agent System Architecture for E-Commerce Applications", Proc. of the fifth International Symposium on Autonomous Decentralized Systems, 2001.

[2] P. Pedreiras and L. Almeida, "The Flexible Time-Triggered (FTT) Paradigm: An Approach to QoS Management in Distributed Real-Time Systems", Proceedings of the 17th International Symposium on Parallel and Distributed Processing. Nice, France 2003.

[3] L. Zhang, "Development Method for Multi-Agent Real-Time Systems", International Journal Of Information Technology, Vol. 12, No. 5, 19-28, 2006.

[4] I. Sommerville, "Software Engineering", China Machine Press, 2006.

[5] A. Perini and A. Susi, "Developing Tools for Agent-Oriented Visual Modeling", G. Lindemann et al. (Eds.): MATES 2004, LNAI3187, pp. 169–182, 2004.

[6] P. Caserta and O. Zendra, "Visualization of the Static aspects of Software: a survey", IEEE transactions on visualization and computer graphics, Vol. 99, 2010.

[7] R. Dodhiawala, N. S. Sridharan, P. Raulefs and C. Pickering, "Real-Time AI Systems: A Definition and An Architecture", Proc. IJCAI, 1990.

[8] C. Duvallet and B. Sadeg, "Des systèmes multi-agents anytime pour la conception des systèmes d'aide à la décision", Technique et Science Informatiques 22(8): 997-1024, 2004.

[9] V. Botti, C. Carrascosa, V. Julian and J. Soler, "The ARTIS Agent Architecture: Modeling Agents in Hard Real-Time Environments", In MAAMAW'99 proceedings, LNAI1647, pp 63-76, Springer-Verlag, 1999.

[10] V. Julian, C. Carrascosa, M. Robello, J. Soler and V. Botti, "SIMBA: an approach for real time multi agents systems", In Proc. of V Conferncia Catalana d'Intel.ligncia Artificial, Castell. Springer-Verlag, 2002.

[11] V. Julian, J. Soler, M. C. Moncho and V. Botti, "Real-Time Multi-Agent System Development and Implementation", CCIA, 2004.

[12] J. Soler, V. Julian, C. Carrascosa and V. Botti: Applying the ARTIS Architecture to Mobile Robot Control, IBERAMIA' 2000, Atibaia, Sao Paulo, Brasil, volume I, pp 359-368. Springer-Verlag, 2000.

[13] R.P. Miguel, V Botti, E. Onaindia, "Formal Modeling Of Dynamic Environments For Real-Time Agents". Multi-agent systems and applications III, CEEMAS 2003, Prague , TCHEQUE REPUBLIQUE, Vol 2591, pp. 475-484, 2003.

[14] T. Marir, F. Mokhati, H. Seridi, "Formalizing ARTIS Agent Model Using RT-Maude", L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 226–231, 2009.

[15] J. Cardoso, "The Syntactic and the Semantic Web", Semantic Web services: theory, tools and application, J. Cardoso (Ed), Information Science Reference, 2007.

[16] T. Abdelaziz, M. Elammari, and R. Unland, "Visualizing a Multiagent-Based Medical Diagnosis System Using a Methodology Based on Use Case Maps", G. Lindemann et al. (Eds.): MATES 2004, LNAI 3187, pp. 198–212, 2004.

[17] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent-Oriented Software Development Methodology. Journal of Autonomous Agent and Multi-Agent Systems, 8(3):203 – 236, May 2004.

[18] T. Marir, F. Mokhati, H. Seridi. Applying Model Checking to ARTIS Model Verification, In Proc. 11th International Conference on Science and Techniques of Automatic control & computer engineering STA'2010, December 19-21, 2010, Tunisia.

[19] B. Leuf, The Semantic Web: Crafting Infrastructure For Agency; John Wiley & Sons, 2006

[20] G. Guerrini, M. Mesiti and I. Sanz, An Overview of Similarity Measures for Clustering XML Documents, Web data management practices: emerging techniques and technologies, A.Vakali and G. Pallis (Eds), 2007.

[21] A. Vakali, G. Pallis, L. Angelis, "Clustering Web Information Sources", Web data management practices: emerging techniques and technologies, A.Vakali and G. Pallis (Eds), 2007.

[22] M. Lotzsch J. Bach, H. Burkhard and M. Jüngel, " Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL". RoboCup 2003: 114-124, 2003

[23] Alexander D. Korzyk, Sr, "Towards XML As A Secure Intelligent Agent Communication Language", 23rd National Information Systems Security Conference (NISSC), 2000.

[24] A. Garro and L. Palopoli "An XML multi-agent System for E-learning and skill Management ", Agent technologies, infrastructures, Tools, and applications, for E-services, Octobre 2002.

[25] M. Wooldridge, "An introduction to multi agent systems", John Wiley & Sons Ltd, 2002.

[26] B. Shneiderman and C. Plaisant, Designing the user interface, Strategies for effective human-computer interaction, 4th Edition. Addison Wesley, 2005.