Tropos For Embedded Real-time Control System Modeling and Simulation

Nesrine Darragi, Philippe Bon, Simon Collart-Dutilleul, El Miloudi El Koursi

Tropos For Embedded Real-time Control System Modeling and Simulation

Nesrine Darragi, Philippe Bon, Simon Collart-Dutilleul, El-Miloudi El-Koursi
Univ Lille Nord de France, IFSTTAR, ESTAS
20 RUE ELISEE RECLUS,
BP 70317, F-59666 VILLENEUVE D'ASCQ, FRANCE
Email: nesrine.darragi@ifsttar.fr, philippe.bon@ifsttar.fr, simon.collart-dutilleul@ifsttar.fr, el-miloudi.el-koursi@ifsttar.fr

***Abstract*—Simulation is the imitation of a system or a process in order to manage the complexity of simulated system or to optimize its performance. This paper presents a agent-based strategy of modeling and simulation. We introduce some modeling methodologies in order to determine the most adequate technique to deal with embedded control systems. We also introduce the Tropos and Agentology methodologies by describing used concepts and how they are integrated with the current stages of Tropos and Multi-agent System methodology. The above is illustrated using an embedded real-time control system as a case study.**

I. Introduction

There are many reasons to make use of formal methods in embedded real-time systems, particularly in railway signalling systems. These includes safety-criticality and complex real-time constraints. The EN50128 guidelines issued by the European Committee for Electrotechnical Standardization is a series of safety requirements for railway control. It contains recommendations based on the criticality, complexity and temporal behavior of the system. It does provide neither an exact procedure nor a unique methodology for the development of embedded critical systems.

Safety is the property which asserts that nothing bad happens in contrast to liveness which asserts that eventually a good thing happens. In our works we focus especially on safety properties and we try to prove that *error* or *fault* states are not reachable for every possible execution.

The aim of our models is to formalize and to anticipate these *dangerous* states which could be categorized as to deadlocked states or erroneous behaviour. It is a sort of future system validation. Starting with the analysis of Functional Requirement Specifications (FRS) which is a set of finite complex or simple sentences describing the behavior of the system, we could capture knowledge divided into two categories. The first one is the domain specific knowledge describing domain properties or assumptions. The second category is the commonsense knowledge which regroups the goals of stakeholders and the system actors.

The paper is organized into sections. After a short introduction, section 2 describes a state of the art of agent-based modeling methodologies. Section 3 is devoted to present the modeling and simulation strategy based on agentology methodology. The next section presents the case study and the application of our approachs.

II. State of the art

An agent-based modeling noted ABM [1] is an approach issued from the artificial intelligence which represents a separate technology. ABM is the opposite of the principle of discrete event or continuous simulation approaches which are destinated to simulate systems by simulating its tasks and *events* using state variables dependencies. Unlike them, ABM simulates systems by defining caracteristics and behaviors of system entities without specifying internal rules. The need of an agent-oriented methodologies is justified by the system complexity as well as the weakness of existing methodologies to support many emergent system characteristics such as dynamicity, cooperation aspects, distribution of software entities and temporal constraints.

Agent-oriented methodology is used to reduce the gap between existing software frameworks dedicated to the implementation of multi-agent systems (MAS) and, more precisely, agent-based simulation and methodologies and guidelines.

The current work is devoted to this approach. We are dealing with systems where we know the caracteristics of every elementary subsystem or component and we are interested specifically in the whole behavior of the system. Agents which are used to simulate these different components communicate and react with each other and with their environment in order to achieve their goals.

We established a comparison study between diverse MAS methodologies in order to choose the most appropriate one. The approach has to cover the development life-cycle of software from analysis to implementation. It is recommended that the chosen methodology heavily based on requirements engineering and influenced by the gaol approach which is our initial conceptual modeling and simulation prerequisite.

In general, modeling methodologies are influenced by approaches such as object oriented approach (Rational Unified Process for example), aspect-oriented approach or GORE [5] for *Goal Oriented Requirement Engineering* like *Keep All Objectives Satisfied* KAOS [10] or *Intentional STrategy Actor Relationships* (i*)[3].Table I shows how methodologies for MAS development are influenced by software development approaches and identified dependencies exist between these methodologies and adapted methods. The table is divided into three parts: MAS methodology, requirements engineering and object-oriented approach. For each methodology we indicate the approach or the method-driven MAS methodology in question.

TABLE I. INFLUENCE OF SOFTWARE DEVELOPMENT APPROACHES ON MULTI-AGENT SYSTEMS METHODOLOGIES

MAS Methodology	Requirements Engineering	Object Oriented
ADELFE [4]		Rational Unified Process
GAIA [1]		Agent-Oriented Methodology
INGENIAS [11]		Rational Unified Process
PASII [12]		Pattern for Agent
SODA [13]		AgentOriented Methodology
Tropos [2]	i* [3]	

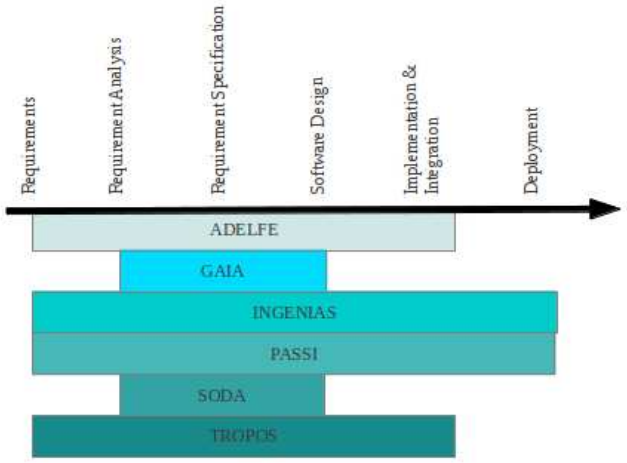

Fig. 1. Phases of the development process covered by different Multi-agent Systems development methodologies

In this work, we need to model and simulate MAS. Figure 1 shows different MAS development methodologies and software development processes they . For example Tropos, ADELFE, INGENIAS [11] or PASSI are used to analyse, to specify, to design and to implement MAS. Tropos methodology [2] responds to our needs in terms of development phases and approaches. It is an agent-oriented methodology for the analysis and design of software from early requirements to late requirements analysis right through to the building of a system model. It is established using goal-based concepts issued from i* and goal-oriented requirements language dedicated to non-functional requirements, Tropos uses other AUML [14] diagrams to deal with MAS aspects. The framework iSTAR or i* defines system agents as entities where relationships are based not only on shared data or informations but also on its common goals, beliefs or abilities.

Entities of Tropos include actors, goals (the strategic interests of agents), capabilities (the ability of an agent to decide the action to execute given its perceptions of its environment and external events), plans (a description of how an agent is able to reach a goal), dependencies (relationships between agents), beliefs (agent's knowledge related to its environment), resources, and finally contributions (a metric to evaluate the relationship between goals, plans or resources specifying if it is beneficial to achieve these goals or not).

Figure 2 shows the Tropos metamodel of the agent concept which inherits from the actor concept. An agent plays the roles and occupies a position. An actor is an agent or a role. An actor has goals and plans. It is dependent on other actors with which it may share resources and goals. The latter is

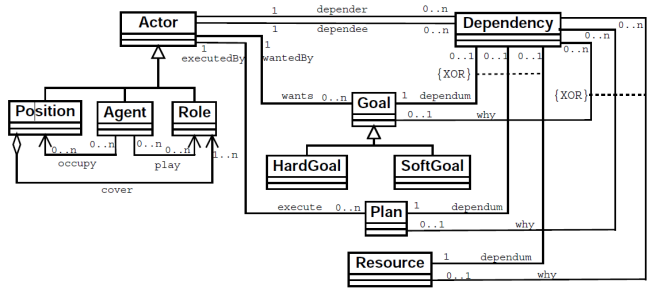

Fig. 2. Tropos metamodel of actor concept

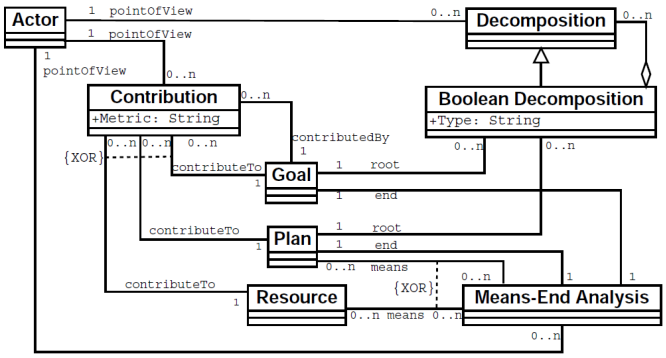

Fig. 3. Tropos metamodel of goal concept

the generalisation of soft and hard goals. The first category is intentions of non-functional requirements in general. The second one is the aim behind the functional requirements.

Figure 3 illustrates the Tropos metamodel of goal concept. An actor has many points of view of its contribution to achieve its goal. This latter may be decomposed by other refined sub-goals in another level of abstraction. A "And" or an "Or" relationships may regroup many sub-goals in order to fulfil the global or the root goal. It has a means or satisfaction.

The methodology of Tropos may be summerized in 5 phases :

- Early Requirement phase : Identification and modeling of stakeholders as social actors. Dependencies of these based on common goals, plans and shared resources are identified.
- Late Requirement phase : Identification of system actors and software agents and the dependencies between them.
- Architectural Design : Definition of subsystem architectures and their connections through data and control flows.
- Detailed Design : Specification of the characteristics and capabilities of each agent.
- Implementation : Implementation of the system detailed design.

III. MODELING AND SIMULATION STRATEGY

Inspired from the Watterfall model [8] but also by best practices contained in several other methodologies ,the agen-

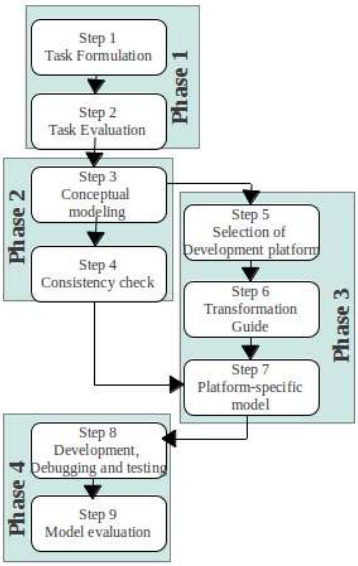

Fig. 4. Process Agentology

tology [6] is a methodology for agent-based modeling. The methodology is independant of any specific technology, programming language or excecution environments. The basic idea of this kind of simulation is Composed of 4 phases and 9 steps, as shown in figure 4. The first phase is the requirement definition wchich consists on formulation and evaluation of requirements or tasks. The second phase is the conceptual modeling. It is composed by two steps; modeling and consistency checking of models.The third phase id the platform-specific modeling which concerns the selection of a development platform in addition to definition of transformation guide and the last step is platform-specific modeling. Finally, simulation modeling which is the fourth pahse. It is dedicated to development, debugging, testing and model evalution. This methodology attempts to offer guidelines to assist the modeling of the system using of one of the most difficult approaches which is agent-paradigm. We will use the ontology as a platform-independent methodology to provide an agent-based model as a means to simulate this model in subsequent steps.

The goal of this work is the simulation of the behavior of subsystems and their interactions under hazardous conditions and to detect possible defects and software faults. We simulate the interaction of subsystems and components considered to be system actors within a real-time execution-like environment. Each one of our actors has specific characteristics such as capabilities, responsibilities, goals, plans and resources which could be shared with other actors. A real-time embedded system is characterized by being driven by real world events. This issue could be simulated by MAS.

First of all, we specify the study case in general and we provide functional requirement specifications (FRS) in the subsequent step. We are dealing with embedded real-time control systems which have some specificities which must be respected.

Conceptual modeling is ”the activity of formally describing some aspects of the physical and social world around us for the purposes of understanding and communication” [Wiki]. The aim of this phase of the simulation project is to transform requirements into system models that are platform-independent and which contain different diagrams of different views. The choice or these conceptual diagrams should be meticulous in order to depict various aspects of the system and cover different views.

First we propose to define an object model to the very high level of granularity which is the abstraction of the real system. In the second step, we create an agent diagram to determine system actors in the defined scope. We propose to be general in this phase and give a global model without detailed design. The next step focuses on the creation of a global model for each agent and the determination of dependencies between them. In the fourth step, a detailed agent model is provided.

IV. CASE STUDY: ERTMS/ ETCS SYSTEM

The FRS of European Rail traffic Management System/ European Train Control System (ERTMS/ ETCS) [9] is the chosen embedded control system to be studied. The aim of this project is to unify the transeuropean railway network with the same and unique train control system. ERTMS is decomposed by on-board equipment which is the embedded system and Trackside equipment which is the static system (Wayside). The architecture of the whole system is illustrated by figure 5.

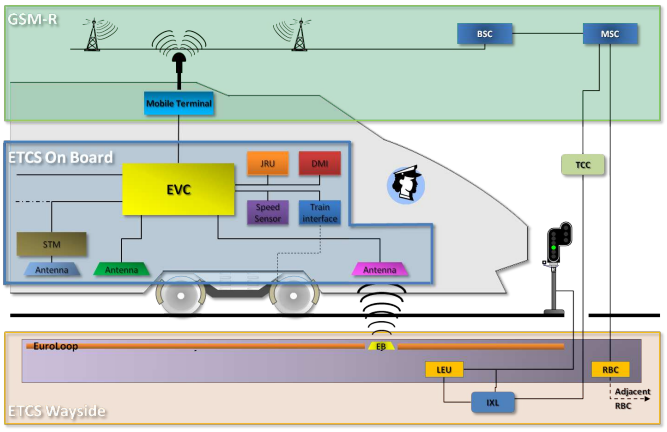

Fig. 5. ERTMS Architecture

We will focus on the Start of Mission (SoM) procedure since it is the first procedure to apply to start the train equipped by ERTMS. We note that in the embedded system is composed of ETCS on-board equipment beside other systems. The procedure ”Start of mission” is totally described in [9]. This procedure is used if the train is awake, if shunting movements are finished, if a mission is ended or if a slave machine becomes a leading engine. In fact, this procedure is used when the on-board system is in Stand-By mode. The procedure describes the status of system data required and details the table of requirements for the procedure.After the table, a complex flowchart describes the interactions between the different components of the system and gives a graphical vision of the procedure. We can notice that on-board system interacts with Radio network and RBC and the flowchart show how they interact between them.

A. System Evaluation for Simulation

The first step of the SoM simulation is to formulate the task which is provided by the subset of the procedure. We analyse the simulation description and we evaluate specifications in order to determine if the present case study is simulable or not by an agent-based model. The aim of this phase is to prove the efficiency of the making use of agent approach and its suitability and convenience in the study case. Therefore, we need to know some general specifications of agents such as making decision ability, the system dynamicity, granularity of the treated system, ect... In the case of SoM procedure , (i) there is one actor that makes decision decision- driver who can make many kinds of decisions. (ii) The system dynamicity is represented by faults avoidance in real-time execution. (iii) The system behavior is treated on a macro level which coincide with the principle of agent-based modeling, contrary to flowchart, activity diagrams or state transition diagrams which are typically viewed in macro-level. (iv) In micro level, each actor has its specificities and we note that there are different used agent architectures. (v) The environment of the MAS is characterised by the presence of spatial factors which may influence the simulation. All these previous metrics mean that an agent-based model is suitable for modeling an embedded real-time control system in general and our sturdy case in particular.

B. Conceptual Model

The concepts of hard and soft goals are used to model functional requirements and quality attributes respectively. Tropos, since it adopts i*, uses these concepts in addition to actors who could be an agent, role or position. Goals can be classified into types or categories. We talk about software and other behavioral goals.

A software goal announces an alternative that should be chosen by the system to reach another goal. A behavioral goal describes the expected behavior of a system. There are two subtypes of behavioral goals: "Achieve/ Cease" and "Maintain/ Avoid." The first class includes goals describing a behavior that must be satisfied or not under conditions in a limited time in the future. The second class is the set of goals describing a behavior that should be satisfactory under conditions over time or behavior should never be present in the system under any condition respectively. We propose the following definition.

Definition 1. Lets S be a set of states, S_i an initial state, S_f a set of states in the future and t_0 is the current instant such that the following hold for $S_i \subseteq S$ and $S_f \subseteq S$:
$F(S_f) \rightarrow \exists t(t{>}t_0 \wedge P(S_f))$
$G(S_f) \rightarrow t(t{>}t_0 \rightarrow P(S_f))$
$Achieve[S_f]$: if $[S_i]$ then $F(S_f)$
$Cease[S_f]$: if $[S_i]$ then $\neg F(S_f)$
$Maintain[S_f]$: if $[S_i]$ then $G(S_f)$
$Avoid[S_f]$: if $[S_i]$ then $\neg G(S_f)$

C. Goal-based reflex agent Architecture

A goal-based agent is an agent having knowledge about its goals and what actions to choose to achieve it. A reflex agent is an agent who maintains an internal state and based in "condition-action" rules it updates its state.

An hybrid architecture is a way to switch between the intuitive reaction and thought deliberation respecting the circumstances and environment events. Moreover, it is the latter that determines the expected behavior of the agent and these characteristics. An agent is considered as an autonomous entity or a decision-maker.

On an abstract view, an hybrid agent is composed of several layers each of which depends on a specific architecture of an agent. Each class of agents has its advantages and disadvantages. We can choose the most suitable functions in architectures using various agents although we face the problems of mutual interference.

The hybrid architectures are in hierarchical layers allowing the boundary delimitation between the different features of each level. The layers of the highest level are reserved for reflection, decision. This is the area of reasoning. The lowest layers are those devoted to communication and perception of the environment. Between these two levels, areas of interaction, knowledge or planning may exist and it depends mainly on the architecture.

Each hybrid agent must have at least two layers of different architectures. The general behavior of the agent will depend concepts of architecture but also the alignment of these architectures. The architecture of our agents as illustrated by figure 6 is hybrid and each actor of our system has the following properties:

- A set of states
- A set of goals
- A knowledge base (KB)
- Capabilities
- Actions

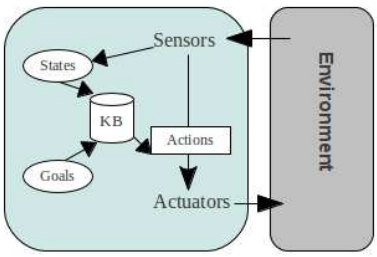

Fig. 6. General Goal-based Reflex Agent Architecture

V. Conclusion

In this work we present a modeling and simulation strategy for embedded control systems based on Tropos and Agentogy. Our approach differs from existing control system simulation approaches in the following way. Current approaches rely mostly on general-purpose simulation modeling paradigms based on continuous, discrete or hybrid simulation techniques. This trend is explained by the high degree of interaction between mechanical and electrical components in embedded control systems. Among others, our contribution is the use of agent-based simulations because these models are often appropriate and interesting to study the behavior of the entire

system where its characteristics are known. This paper is a first attempt to make ERTMS /ETCS modeling and simulation more specifically goals oriented. Future work is needed to validate these models in practice and to propose a detailed model and architectures for subsequent steps of simulations.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their helpful and constructive comments and suggestions on this research.

REFERENCES

[1] W. Michael and N. R. Jennings, *The Gaia Methodology for Agent-Oriented Analysis and Design*,In: Autonomous Agents and Multi-Agent System, 2000.

[2] J. Castro, M. Kolp and J. Mylopoulos *A requirement-driven development Methodology*,In: Lecture Notes in Computer Science, pages 108-123. Springer, 2001

[3] E. Yu *Towards Modeling and Resoning Support for Early-Phase Requirement Engineering*,In: Proceedings of the 3th IEEE International Symposium on Requirements Engineering, Washington, 1997.

[4] C. Bernon, M-P, Gleizes S. Peyruqueou and G. Picard *ADELFE: a Methodology for Adaptive Multi-Agent Systems Engineering*,Third International Workshop "Engineering Societies in the Agents World", Madrid, 2002.

[5] A.Van Lamsweerde, *Requirements Engineering, From System Goals to UML Models to Software Specifications*,Wiley, 2009.

[6] T. Salamon, *Design of Agent-based Models*, Academic series, 2011

[7] D. Bertolini, A. Perini, A. Susi and H. Mouratidis. *The Tropos visual modeling language. A MOF 1.4 compliant meta-model.* 2004

[8] W. Royce, *Managing the Development of Large Software Systems*, Proceedings of IEEE WESCON 26 (August): 19, 1970

[9] UNISIG, ERTMS Users Group , *Functional System Requirements Specification, FRS*, FRS V4.29, 1999

[10] A. Dardenne, A. V. Lamsweerde, S. Fickas *Goal-directed requirements acquisition*, In: Science Computer Program, vol. 20, page 3-50, April 1993

[11] J. Pavon, J. Gomez-sanz *Agent Oriented Software Engineering with INGENIAS*, In: Third International central and Eastern European Conference on Multi-Agent Systems (CEEMAS), Springer verlag, 2003

[12] P. Burrafato and M. Cossentino *Designing a Multi-Agent Solution for a Bookstore with the PASSI Methodology*, In: AOIS'02 at CAiSE'02, Toronto, May 2002

[13] A. Omicini, *SODA: Societies and infrastructures in the analysis and design of agent-based systems*, In: Agent Software Engineering, vol. 1975 of LNCS Springer, 2001

[14] A. Bauer, J.P. Muller, J. Odell, *Agent UML: A Formalism for Specifying*, In: Multiagent Software Systems Proceedings, ICSE 2000 Workshop on Agent-Oriented Software Engineering AOSE. Limerick, Springer Verlag, page 121-140, 2000