Verification of Temporal Requirements of Complex Systems Using UML Patterns, Application to a Railway Control Example

Mohamed GHAZEL *, Malek MASMOUDI * and Armand TOGUYENI †

*INRETS - The French National Institute for Transport and Safety Research

Villeneuve d'Ascq, France

† LAGIS - Laboratoire d'Automatique, Génie Informatique et Signal, Ecole Centrale de Lille

Contact: mohamed.ghazel@inrets.fr

***Abstract*—Temporal aspects have a vital importance while dealing with the verification of critical systems. Time constraints may reflect both security and performance requirements. Thereby, verifying the temporal requirements is a major task in the validation of critical systems. In this paper, we discuss a new approach for the specification of the temporal requirements within complex systems. We also sketch a global verification method integrating the specification process proposed.**

The specification is made in a systematic way on the basis of some generic patterns we developed. These patterns are designed starting from a classification of temporal requirements that we have established while trying to cover at best all the usual requirements one may encounter while dealing with the verification of complex systems. The verification process of a given system is performed using observers instantiated from the proper patterns of the requirements identified.

Unlike several existing approaches, our approach proposes means to assist the analyst in the requirements' specification step. Moreover, it allows for the verification of various requirements at once. A use case study from the railway operation field allows the illustration of the various concepts discussed.

***Index Terms*—Verification, complex systems, temporal requirements, checking, patterns, observers, dependability, UML, Stocharts, railway control.**

I. Introduction and context

Complex systems are characterized by a large number of components of various kinds (mechanical, electrical, computer ...) that have different types of interactions (local, simultaneous ...) which explains the complexity as regards the predictability of their behaviour.

The background of the study is the evaluation of complex systems and more particularly the checking of temporal requirements in the field of dependability and interoperability. The problem of interoperability arises mainly as components for the construction of a system come from different sources. A typical example is ERTMS (European Rail Traffic Management System) [ERT-web], the new common control-command and signalling system for the European railway system. This type of system is defined by international standards. The differences in the interpretation of specifications by the manufacturers of components that are integrated to build a system are one of the main causes of interoperability problems. This causes difficulties for manufacturers of systems to integrate components from different manufacturers.

Temporal requirement can reflect different aspects in terms of dependability: availability, safety and so on. These requirements can be qualitative or quantitative. An example of qualitative temporal requirement in railway system can be "Doors should not open before the train stops". A quantitative temporal requirement is more explicit about the time factor as illustrated by the following example: "The doors are open 5 seconds after stopping the train". The checking of such requirements is essential especially for critical systems. In these systems and more particularly in the field of transport, non satisfaction of time requirements may cause harm to humans and/or material (e.g. loss of life). That is why the detection of errors as well as the design or validation stages is becoming highly recommended and essential. This can be achieved through methods of formal checking.

The first researches on the assessment of temporal requirements come from the areas of network protocols [Stef93] and multimedia applications [Wahl94] because of the importance of time in these areas. The results of these works imply other works on the dependability of complex systems in general and manufacturing systems and transportation systems in particular.

The objective of the study presented in this paper is to set the foundations for a generic approach for the checking of temporal requirements of complex systems. The final goal is to develop software tools in order to implement the methodology.

The paper is structured as follows: In section 2, the approach will be presented. First, we propose a classification of temporal requirements that we have achieved. In order to ensure genericity, checking patterns will be developed and examples will be given. Finally, an illustration of the approach pertinent to control-command in the railway system is presented. In Section 3, a global view of the implementation of the approach will be proposed. We conclude this study in Section 4 by presenting the perspectives of this work.

II. THE DEVELOPED APPROACH

The main approaches of checking temporal requirements are based on the analysis of accessibility of formal models [Fon08], which are of two types:

- The audit by model checking [Cous05] which is to use a checker tool to browse a formal model of the system and check if it fulfils a temporal requirement specified in temporal logic;
- The checking based on observers [Ghazel09] where these observers are entities expressed in the same language as the model of the system. They interact with the behavioural model of the system without implying any perturbation. The interactions enable it to observe and verify the specified temporal requirements of the system.

Because of our interest in interoperability between components of a complex system we have chosen to base our verification approach on observer techniques. Indeed, the important advantage of observers is that they evolve according to signals received from a system or its model. To build them, it would not require the knowledge of the system's internal behaviour. The observers capture the behaviour of the component from an external point of view which is essential for interoperability.

The literature describes some tools which implement methods of checking by observation. GOAL is an example of an industrial tool that implements a checking method [Dol03]. However, no procedure has been proposed for the construction of the observers.

A key idea of this work is to develop generic models to check temporal requirements without the need to know a priori the internal behaviour of the system. Two types of methods give the possibility to reach this goal:

- The evaluation of an existing system for example to assign a certification to this system: in this case, the checking process considers the system as a black box. Specifically, the checking process will be based on observable events generated by the system. It will use predefined inference mechanisms;
- The evaluation of a behavioural model of a system in process of being designed: the developed checkers will be added to the model of the system. They will detect possible violations of requirements to bring the user to reconsider its design. In this case, the check is done by simulation.

A. Approach for the patterns' development

In this section, we outline the approach we have adopted for the development of patterns on classes of temporal requirements. It can be divided into three stages (see Figure 1):

- The first stage has studied requirements which implicitly or explicitly handle the time factor. The goal is to identify these requirements and to define their characteristics;
- Then, without being exhaustive, we conducted a hierarchical classification of temporal requirements according to the criteria we set later in this paper. The result is a typology which will be presented in the form of a UML class diagram with inheritance relations;
- Finally, the third stage develops for each type of requirement a standard and generic brick called "pattern", which makes it possible to obtain a checker for this requirement.

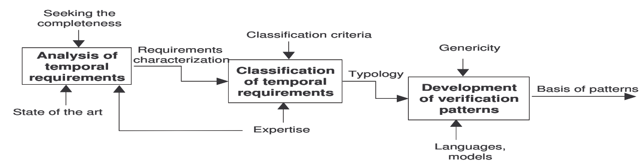


Fig. 1. SADT Diagram of the Patterns' Development Method

B. Development of a typology of requirements

In literature, there is no rigorous classification of temporal requirements that may apply to different processes of a given application. A process represents an ordered sequence of punctual actions $(a_1, ..a_i, ..a_n)$. Our work has started with the study of RT-LOTOS operators [Sad07] to establish an initial classification. It distinguishes between the quantitative temporal requirements that explicitly handle the time factor and qualitative requirements where temporal requirements are implicit. The classification is presented below on a UML class diagram (see Figure 2). The legacy relations between classes can define a hierarchy of classification criteria. OCL constraints have been added to specify certain aspects of classification and to ensure its consistency. Finally, 11 types of requirements have been identified. The classification may be enriched if new sets of requirements are identified. The requirements outlined in the proposed classification express temporal constraints defined with regard to a relative time. It remains that, among other things, one should take into account the temporal constraints expressed in absolute time. For example, a requirement that *"The X system starts the 01/01/2009 at 00h00mn01sec"* is a quantitative requirement expressed in absolute time.

C. Construction of patterns

1) Main idea: A pattern is a ground for a graphics program. This concept comes from architecture and is transposed to software engineering for object-oriented programming [Gam07]. The basic idea behind the use of patterns is mainly to promote the re-usability.

Our approach aims to develop patterns instantiation which would get checkers for the temporal requirements of a given system. For this, a pattern is designed for each type of requirement. The rest of the paper will show the form that our patterns will take and it will also present some examples of instantiation.

As the checking process should not influence the behaviour of the system, the patterns are designed so that their bodies are external blocks operating in parallel to the behaviour of

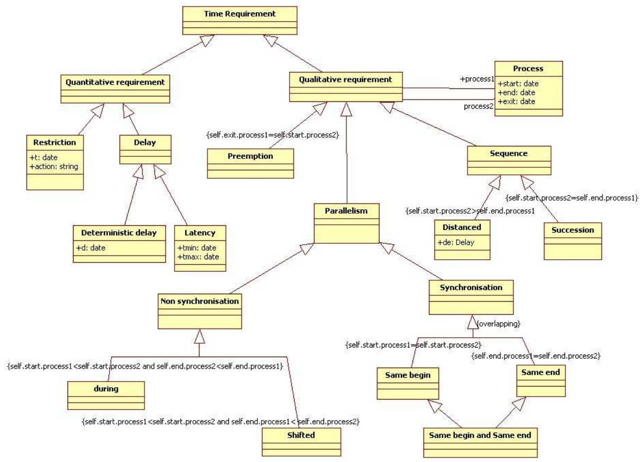


Fig. 2. UML Class Diagram for the Classification of Temporal Requirements

the system. In other words, the role of observer checkers is to monitor the violations of pre-defined temporal requirements for a given system but not deal with these violations. Thus, for an existing system which has no behavioural model, our idea is to ignore the internal behaviour of the system. Practically, this means that the designer will not try to model the system. This approach is justified by the fact that one is interested here in the checking of the interoperability of complex systems as announced earlier in the paper. The construction of separated and independent checkers for the requirements can solve a major problem which is often mentioned in the checking/validation approach. This problem is the interdependence of the verification process.

2) Stochart language: To develop our approach, we have adopted the Stochart language. It has been proposed by Jansen [Jans03] as an UML extension of Statecharts. The Statecharts themselves have been developed by Harel [Har87] as an extension of state automata (Moore Machine [Moor56]). Statecharts extend these finite state automata (FSA) with the concepts of parallelism and hierarchy. Stocharts extend Statechart by introducing the concepts of choice and probabilistic stochastic time. The choice of Stocharts in our study instead of other state-transition languages is justified by their additional capacity of expression.

3) Some examples: While developing our patterns, some "reference states" have been set up. These states make it possible to point out some particular facts within the behaviour of the system under check (event occurrence) or some specific dates to which the verification process has to refer. Then, a reference state allows the verification procedure to refer to some landmarks in the system's behaviour.

For an existing system for which we do not have a behavioural model, defining an external reference state dispenses us of seeking to recognize the internal state of the system. On the other hand, when dealing with a design model under validation, having a reference state peculiar to the verification, prevents the verification process from influencing the system behaviour.

We will give hereafter two examples of patterns. The first deals with a quantitative requirement, a Latency. The second deals with a qualitative requirement, a synchronization. For quantitative requirements such as latency, deterministic and non-deterministic delays, by definition, the time constraints considered are quantified. Let us recall that, by *process* we designate a sequence of instantaneous actions. A Latency requirement expresses the fact that a given action a has to occur in an interval [0,t[from a given fact, which will be noted C/ here.

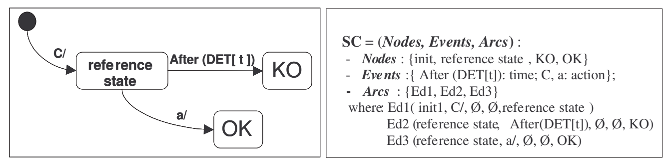


Fig. 3. Verification Pattern of a Latency

The reference state of the pattern is reached through the occurrence of C/ (cf. figure 3). The normal behaviour of the system corresponds to the system state switching from the reference state to the OK state when a is detected after a delay less than t t.u. State KO corresponds to the requirement violation. It is reached if the system stays in the reference state during $t\ tu$ without a having been detected. This is specified by $DET[t]$ function, indicating that a deterministic delay t conditions the switch from the reference state to KO.

Qualitative time requirements (pre-emption, parallelism, synchronization and sequence) do not consider determined time quantities. We will present here the verification pattern relative to a synchronization requirement. For that, we will first define two processes P_1 $[a_1, ..., a_n]$ and P_2 $[b_1, ..., b_m]$, a_i being the ordered actions of P_1, and bj those of P_2. A synchronization expresses the fact that the two processes P_1 and P_2 synchronize at some determined actions S_1? $\{a_1, ..., a_n\}$ and S_2 ? $\{b_1, ..., b_m\}$. In other terms, actions S_1 of P_1 and S_2 of P_2 have to occur simultaneously (cf. Fig. 4).

The normal behaviour of the system is characterized by a simultaneous passage from states A and B into state OK. The KO failing state is reached $\underline{iff}$ states A and B are not reached at the same time, i.e. if actions S_1 or S_2 have been executed one before the other and not simultaneously.

In practice, verifying the set of temporal requirements of a given system is done while instancing our patterns in such a way as to obtain some Stochart model parts which implement the observation mechanisms suitable to the requirements we want to verify. In order to respect the genericity of our patterns, a special care is needed while establishing them. In total, 9 patterns have been developed.

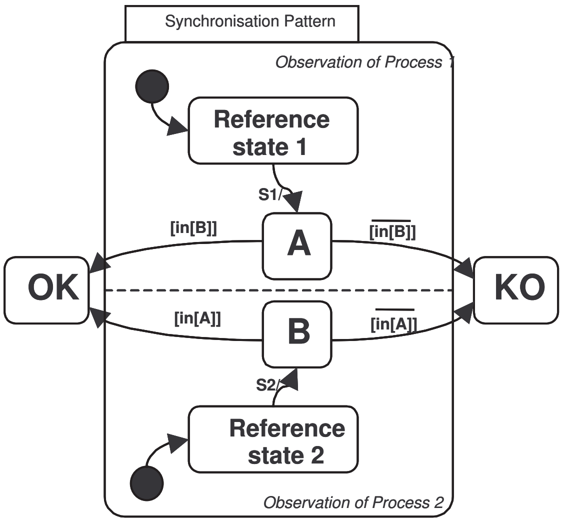


Fig. 4. Verification Pattern of a Synchronization

D. Application

In this section, an illustration of a Level Crossing control system will be discussed. A Level Crossing (LC) is an intersection between a railway and a road path at the same level. In France, as in the majority of countries, under nominal circumstances, railway traffic has an absolute priority while passing the level crossing. The level crossing's automatic control system is responsible for closing and opening the LC for road traffic. It generally compounds protection barriers, road signalling lights and sound alarms. The example discussed here is inspired from the study presented in [Alur92] where a rough description of the system operation has been proposed.

The system is composed of three modules: the **Train**, the **Control Centre** and the **Barriers**. Here, we make an abstraction of the other components of the protection system(road signalisation, alarms). These modules operate in an interdependent way, and communicate thanks to some synchronization events: $approach$, $leave$, $lower$ and $down$. The global operation of the system is depicted in Figure 5, with a StoChart diagram.

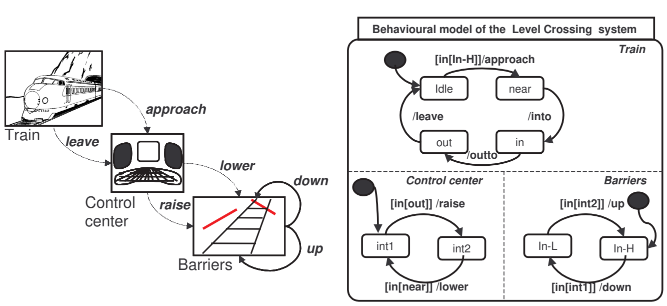
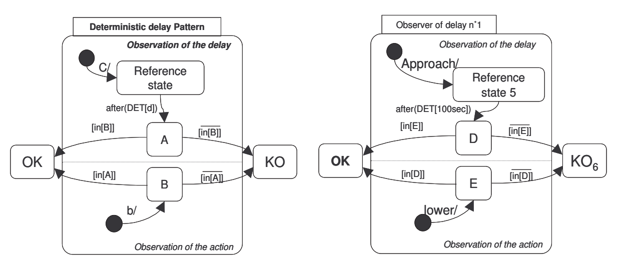


Fig. 5. Illustrative Figure, Behavioural Model of the Level Crossing System

The precise description of the system operation which integrates the temporal aspects, is as follows: when the train is approaching the crossing, it sends an $approach$ signal to the control centre. Then, the train enters the crossing zone of the LC at least 300 seconds later. When the train leaves the crossing, it sends to the control centre a $leave$ signal. $leave$ is sent within 500 sec after the $approach$ signal. The control centre sends a $lower$ signal to the barriers, exactly 100 sec after receiving the $approach$ signal, and sends a $raise$ signal within 100 sec after the reception of the $leave$ signal. The barriers' system responds to the $leave$ signal with a $down$ action within 100 sec, and responds with an up action to the $raise$ signal between 100 and 200 sec later. Let us now extract the various temporal requirements the system has to satisfy. From the description given above, six temporal requirements have been identified:

1) a Latency of 100 sec between the sending of the $approach$ signal and that of the $leave$ signal;
2) a parallelism between actions $into$, $outto$ and $leave$;
3) a latency of 100 sec between the $leave$ signal and the $raise$ signal;
4) a deterministic delay of 100 sec between the $raise$ signal and the execution of the up action;
5) a latency of 100 sec between the detection of the $lower$ signal and the execution of the $down$ action;
6) a deterministic delay of 100 sec between the $approach$ signal and the $lower$ signal.

Once these requirements are identified, the next step consists in developing suitable observers for our requirements; these observers are obtained by instancing the adequate patterns. This instantiation is made while replacing the patterns' attributes by the requirements' parameters. By this way, the observers' establishment is made in a systematic way. In figure 6, we propose the example of instantiation of the 'deterministic delay' pattern in order to establish the observer for the sixth requirement.



(a) Verification pattern of a deterministic delay requirement (b) Observer of the deterministic delay 100sec between Appro ach et Lower

Fig. 6. Pattern and Observer (instance) for a Deterministic Delay

Note that in order to distinguish between the observers established for the various requirements, some indices have been added to the reference states as well as to the corresponding KO states (KO_i for the violation of $requirement_i$). Also, we distinguish between the execution of an action (/action) that we may find in the behavioural model of the system, and the detection of the corresponding event (action/) that we may find in the observers' models. The observers established for the six identified temporal requirements are integrated as a new process running in parallel with the behavioural model of the system in an AND stochart node (cf. figure 7).

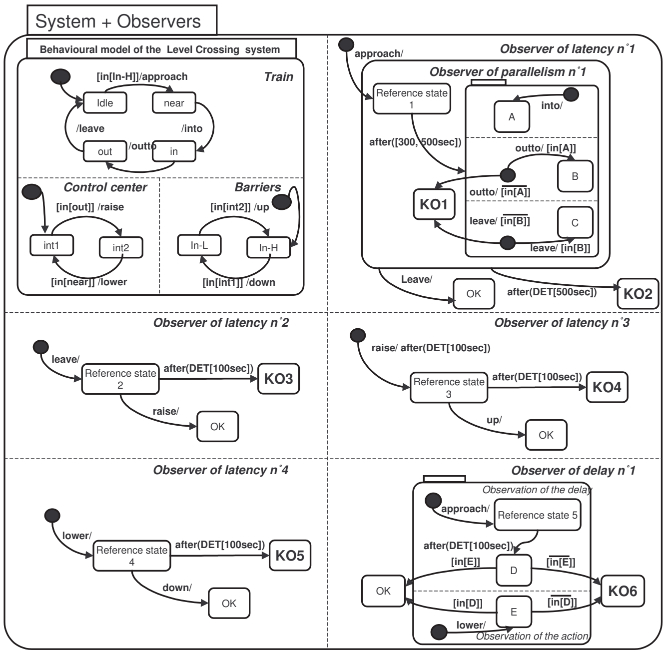


Fig. 7. Complete Model

The requirements' verification is thus done by "catching" the events produced by the system which are depicted in the behavioural model. Violating a given $requirement_i$ will have as a consequence the reach of the KO_i state of the corresponding observer.

Generally, the verification process of a given system is made in a different manner according to the system we are dealing with: if we have to verify a system while being designed for which we know the internal behaviour (model), the verification process is made by simulating the model obtained after having integrated the verification instances with the behavioural model of the system. Concretely, one proceeds by analysing the simulation traces. For the second case, if we deal with a physical implementation of some given specifications for which we do not know the internal behaviour, the verification is made on the basis of the observable events generated by the system, while using inference mechanisms implemented in some verification models (cf. figure 8). The LC case study discussed above corresponds to the second situation.

III. Proposition of Implementation

In this section, an implementation of the entire approach is proposed (see Figure9). The proposed implementation differentiates between the approach's two different cases of

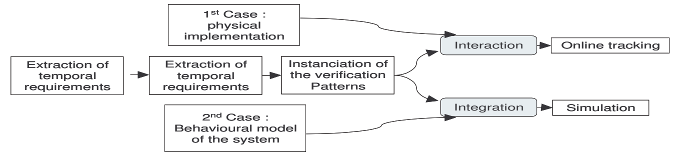


Fig. 8. Setup of the Approach According to the System to be Verified

application, namely the checking of an existing system and that of a design model. The checking process is represented by dotted arrows for the first case, and with bold arrows for the second case.

In the first case, the problem consists in designing a tool that, on one hand, integrates checkers of temporal requirements and, on the other side includes an interface capable of capturing events generated by the system to be verified. These events will excite checker models indicating possible violations of pre-defined requirements.

Regarding the second case, the authors propose in [Herm05] a detailed procedure that will inspire us to provide a complete implementation in terms of software tools and languages for simulation. The idea here is to translate the obtained Stocharts models in MoDest language [D'Arg01], a formal language that describes timed systems. Some promising works already exist on the automatic processing from Stocharts to MoDest [Herm05]. This transformation of models is implemented by the pair of tools Motor-Mbius [Deav01], which allows for the simulation of discrete event systems with generation of reachability traces (see Figure9). The Mobius tool analyzes the traces obtained by simulation based on scripts that give the result of the checking process. The tool also gives illustrations and contrary examples to help the user understand the errors of its design.

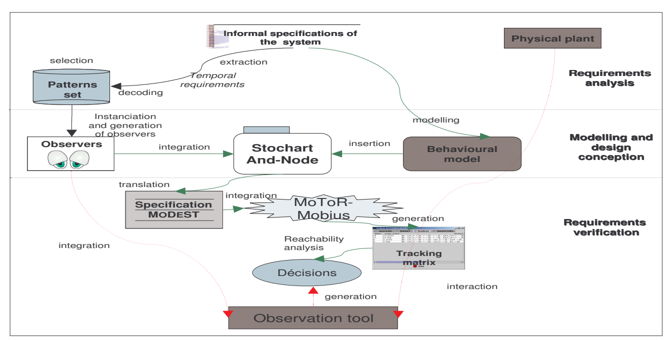


Fig. 9. Proposition of a Complete Implementation for the Developed Approach

The checking process presented in Figure 9 can be broken down into three main stages:

- A first stage of processing with requirements identification and instantiation of the appropriate pattern. This step is common to both evaluation cases;

- A second stage consists in integrating the observers in the checker tool (for analyzing the behaviour of the system from an external point of view), or respectively with the behavioural model of the system (to use simulation to evaluate a design model);
- A final checking and analysis of observable events of the real system, or the analysis of reachability traces generated by the pair of tools Motor-Mobius.

IV. Conclusion and Future work

In this paper, we have first proposed a classification of the most common temporal requirements. Of course, our classification could be enriched in the future. But, based on the data we have found so far in the literature, we tried to make it as comprehensive and coherent as possible.
After that, checking patterns for these requirements have been developed. The use of patterns has enabled us to maintain a sufficient level of abstraction to ensure the generic approach. The use of checking has its whole interest in the validation of complex systems. Indeed, in the absence of the behavioural model of a specific existing system, we are exempt from having to analyse and "understand" the system to establish its internal behavioural model. Stochart language has been used for its rich semantics and its extensive capabilities of expression, compared to other modelling languages of discrete event systems.

In terms of perspectives, there remains a substantial work to be done on the extraction of requirements. Indeed, this task is manual and is mainly based on specifications written in natural language and is, therefore, subject to different interpretations. The introduction of standardization in this area would be interesting and would make this task systematic or automated. Moreover, in terms of software tools to support the method, it is necessary to implement interfaces for the integration of other models than Stocharts. Finally, a formalization work is still necessary to theoretically validate our approach [Ghaz07]. In some areas such as the control-command in the railway, the use of formal evidences is required in the certification process of critical equipments such as interlockings, embedded control systems, etc.

References

[Alur92] Alur, R., Courcoubetis, C., Halbwachs, N., Wong-Toi, H., *An implementation of three algorithms for timing verification based on automata emptiness*, 13th Real-time Systems Symposium, pp157-166, Phoenix, USA, 1992.

[Cous05] Cousot, R., *Verification, Model Checking, And Abstract Interpretation*, 6th International Conference, Vmcai 2005, Paris, France, January 17-19, 2005.

[D'Arg01] D'Argenio, P. R., Hermanns, H., Katoen, J.-P., Klaren, R., *MoDeST: a modelling and description language for stochastic timed systems*, Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods. Performance Modelling and Verification: PAPM-PROBMIV, Vol-2165 of LNCS, Springer, pp87-104, Berlin, 2001.

[Deav01] Deavours, D. D., Sanders, W. H., *Mobius : Framework and atomic models*, Proc. 9th International Workshop on Petri Nets and Performance Models (PNPM '01), pp 251-260. IEEE, 2001.

[Dol03] Doldi, L., *Validation of Communications Systems with SDL*, Wiley, 2003.

[ERT-web] *http://www.ertms.com/*

[Gam07] Gamma, E. et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, 37th edition, ISBN 0-201-63361-2, Addison-Wesley Professional Computing Series Ed., 2007.

[Fon08] Fontan, B., *Méthodologie de conception de systèmes temps réel et distribués en contexte UML/SysML*, Phd Thesis, université de Toulouse, France, janvier 2008.

[Ghaz07] Ghazel, M., El-Koursi, E.M., *Automatic Level Crossings: From Informal Functional Requirements' Specifications to the Control Model Design*, Proceedings of IEEE-SoSE2007, Texas-USA, 16th - 18th April 2007.

[Ghazel09] Ghazel, M., Toguyéni, A. and Yim, P., *State Observer for DES under Partial Observation with Time Petri Nets*, Journal of Discrete Event Dynamic Systems: Theory and Application", Springer Ed., Vol. 19, Issue 2, pp. 137-165, 2009.

[Har87] Harel, D., *Statecharts: a visual formalism for complex systems, Science of Computer Programming*, Vol-8, No3, 231-274, 1987.

[Herm05] Hermanns, H., Jansen, D. N., Usenko, Y. S., *A Comparative Reliability Analysis of ETCS Train Radio Communications*, REPORTS of FB/TR 14 AVACS - Automatic Verification and Analysis of Complex Systems-, No2, février 2005.

[Jans03] Jansen, D. N., *Extensions of Statecharts with Probablity, Time, and Stochastic Timing*, PhD Thesis, University of Twente, 2003.

[Mas08] Masmoudi, M., *Evaluation des systèmes complexes : Application au contrle commande ferroviaire*, Rapport de Master de recherche, INRETS-LAGIS, Septembre 2008.

[Moor56] Moore, E. F., *Gedanken-experiments on Sequential Machines*, Automata Studies, Annals of Mathematical Studies, Princeton University Press, Princeton, N.J No-34, pp129-153, 1956.

[Sad07] Sadani, T., *Vers l'utilisation des réseaux de Petri temporels étendus pour la vérification de systèmes temps-réel décrits en RT-LOTOS*, PhD Thesis, Institut National Polytechnique de Toulouse, Mai 2007.

[Stef93] J. B. Stefani, J. B., *Computational Aspects of QoS in an object-based, distributed systems architecture*, 3rd International Workshop on Responsive Comuter Systems, Lincoln, NH, USA, Septembre 1993.

[Wahl94] Wahl, T., Rotherme, K., *Representing Time in Multimedia Systems*, International Conference on Multimedia Computing and Systems (ICMCS'94), pp.538-543, Boston, USA, 1994.