Performance assessment of industrial control system during pre-sales uncertain context using automatic Colored Petri Nets model generation

Moulaye Aa Ndiaye, Jean-François Pétin, Jacques Camerini, Jean-Philippe Georges

Performance assessment of industrial control system during pre-sales uncertain context using automatic Colored Petri Nets model generation.

Moulaye A.A. NDIAYE[1,2,3], Jean-François PETIN[2,3] Jacques CAMERINI[1] and Jean Philippe GEORGES[2,3]

[1] Schneider Electric - Process automation
06510, Carros, France
moulaye.ndiaye@schnedier-electric.com
jacques.camerini@schnedier-electric.com

[2]Université de Lorraine, CRAN, UMR 7039, Campus sciences, BP 70239, Vandoeuvre-lès-Nancy, 54506, France
[3]CNRS, CRAN, UMR 7039, France
moulaye-abdoul-aziz.ndiaye@univ-lorraine.fr
jean-philippe.georges@univ-lorraine.fr
jean-francois.petin@univ-lorraine.fr

***Abstract*—Industrial control systems (ICS) are defined with hardware and software components dedicated to control and monitoring tasks for factory process. Proper functioning of ICS architectures is mainly linked to the performance they offer. System integrators (SI) must know performance of the architecture they propose to the customers by assessing them. Many methodologies have already proven their capabilities to assess ICS architecture performance. One of them is colored Petri Nets (CPN). To assess the performance of ICS architecture using CPN involve defining manually the model. Pre-sales uncertain context involves problematic making this manual model definition challenging.**

This paper introduces a concept allowing automatic CPN model generation by instantiation and parameterization. However before introducing the concept, the paper shows the problematic involved by the pre-sales context. Then shows why CPN methodology is a relevant solution for assessing the performance of ICS in this context.

Keywords—Colored Petri Nets, Industrial control system; performance assessment; automatic model generation; pre-sales

I. Introduction

Performance of ICS architectures are defined through 3 mains criteria. The first criterion is temporal performance gathering all the response time of an architecture and are defined by the time from the occurrence of an input to a response of an output. The second criterion is the programmable logic controller (PLC) scan time representing the amount of time PLC takes to perform its entire automation task based on the process and external inputs [1]. Finally the last criterion is the performance related to critical components load. During pre-sales phase, system integrators (SI) propose and evaluate several ICS architectures that fulfill the customer requirements. However, pre-sales context is characterized by low information about the project, short time range to submit an offer, and limited financial resources and manpower. To assess performance, SI are using currently 3 solutions. The first one consists in designing the architecture on the basis of manufacturers' reference architectures that provide some guarantee about their theoretical behavior and performances. The main advantage of this solution is reducing time and resources for submitting a commercial offer. However when the architecture is deployed, some gaps are noticed between expected and implemented performance that lead to architecture modifications, cost and penalties for the SI. The second solution is to oversize architecture critical components which can lead to performance bottlenecks. Even if the performances will be guaranteed, this solution represents a real commercial risk due to the high cost of the architecture. The third solution is to test the future architecture or its main parts in a laboratory. This solution has double guarantees on its performance and on the technical pertinence of the commercial offer. The main disadvantage of this solution is the investment in term of financial and manpower without any warranty on winning the project.

The limits of theses solutions inspire Schneider Electric to develop a new solution able to assess the performance of the ICS architecture during pre-sales, in order to secure a commercial offer. The objective is to provide a software tool, which will be able to model, simulate and assess performances of the several and various ICS architectures without engaging huge time, human and financial resources.

This paper will firstly make a state of the art in section 2 about the pre-sales context and the problematic involve by this context. In the section 3 a proposition is made in order to assess the performance in this pre-sales context using CPN model. In the section 4 an example of configuration is shown in order to explain the proposition. Finally the section 6 concludes and gives direction for future works.

II. Requirements

A. Engineering process requirements

To define an ICS architecture during pre-sales stage, information known by the SI are the Input / Output (I/O) list, the P&ID diagram, the cycle time of all periodic task and the

performance specifications. With these limited information, prior choices must be done by the SI. These choices are made based on his knowledge and experience in term of topology and architecture components. At this stage, he does not have an absolute certainty regarding the validity of these choices compare to performance specified by the customers.

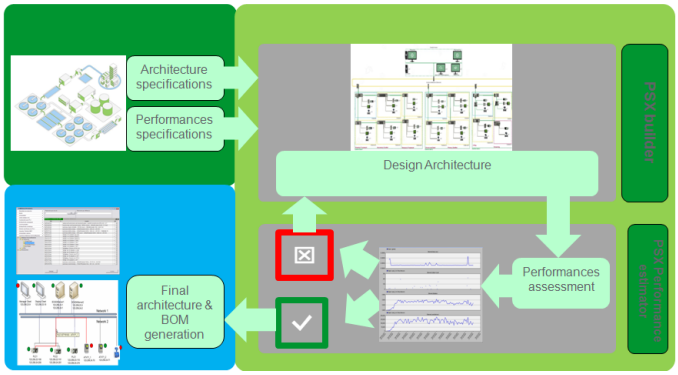

Fig. 1 ICS Architecture definition workflow

As shown on Fig. 1, the validation of the architecture depends on output performance. If the assessed performances correspond to performance required by the customers then the architecture is considered as validate and the SI can submit a build of material and an offer. If the performance does not match with performance specifications, he will modify the architecture and run other simulations repeatedly. So iteration between the architecture design and the performance assessment is done until the relevant architecture is found. Thereby SI must test a wide range of architecture, check the performance and modify it based on the output performance. He has to work with trial and error methodology in order to reduce this uncertainty [2] and find the relevant architecture based on the required performance.

Main requirement about the engineering process can be formulated as following. System integrators need a software tool that is able to compute ISC architecture performance. This tool will be based on a formal description of the architecture behavior. However, the modeling activity has to be as most, as possible, automatized to be compliant with the available resources during the pre-sales stage. In other words, the ICS model is expected to be automatically generated from the informal description of the ICS architecture in terms of topology and component performance. Objective is to avoid manual definition of the ICS models, which is time consuming.

B. *Modeling requirements*

Due to the variety of ICS network topologies and the numbers of connected equipment, the formalism used for the ICS modeling has to face the following requirements:

- The formalism has to support a hierarchical representation of the ICS: high level model representing the ICS topologies, definition of generic models for all ICS components,
- The formalism has to support the modularity and reusability of models,
- The formalism must support the characterization of timed behavior,
- The formalism must enable the characterization of random input parameters with probability laws. Indeed, some parameters characterizing the automation process and the component behavior [3] are completely random [4] when they depend on user actions, components failures or uncertainty about response time of the devices; moreover, these parameters are often described with non exponential distribution law (uniform distribution, Poisson or Weibull)

C. *Scientific positioning*

Performance assessments of ICS architecture have been studied during these past years. Many methodologies have been developed. This section presents a brief state of the art of available method trough two main families: analytic evaluation and simulation.

Analytical evaluation consists of using analytical models to compute the delay introduced by ICS components. Analytic approaches often provide the temporal performance in terms of a maximal time delay between components. This is the case of the network calculus which is a determinist theory of queuing systems found on computer network [5] [6], the max + algebra [7] or even the model checking [8] which is based on the formal verification of time communication between components. Other approaches rely on probabilistic models, such as the Markov Chain theory [9], which provide analytically an estimation of the response time if the system models are homogeneous to a Markov or Semi-Markov processes (mainly when only using exponential distributions). This limitation, with regard to the probabilistic features of the ICS requirements, makes this method unsuited for this context.

The second method is the simulation which consists of model state space exploration. This method is done by modeling the functional behavior of component and it is also adapted to complex system where synchronous, asynchronous and time delay notion are important.

Among many methodologies that have been studied, Colored Petri Nets (CPN) has already proven its efficiency of modeling and assessing performance of ICS. For instance [8] [7] have developed a model of a distributed architecture using CPN methodology. Specifics model have been designed for devices like PLC, whereas network performances have been retrieved from experimental benchmark studies. In [10], the modeling is extended to specific network devices (Ethernet switches) by using High level Petri Nets (HLPN) formalism. Nevertheless in these approaches, CPN models have been specifically defined manually for a given ICS architecture.

The CPN and the embedded concepts of "color", "hierarchy" and "time" are well suited to cope with those requirements [11], [12]. Moreover, this formalism enables stochastic timed transitions with enabled memory policy for firing transitions. At last, using non exponential distributions in the ICS model leads to promote the use of Monte-Carlo

simulation to assess the ICS temporal performance. For all these reasons, CPN appears to be an efficient choice for the pre-sales software tool to be developed. The work is supported by CPN tool [12] for modeling, simulation and performance assessment. Thereby functions described below use tool formalism through the Standard ML programming language [13].

III. Proposal

This section consists of two sub sections and presents the methodology to assess performance of ICS architecture during pre-sales.

A. Generic ISC architecture modeling

The manual CPN model definition of ICS architecture is complex and time consuming due to the variety of network topologies and to the number of connected components on a topology. However, despite network topologies variety and process type, ICS architectures involve the same families of components (SCADA, PLC, I/O devices, network, …) [14]. Starting from this statement, a generic modeling approach can be proposed.

A generic hierarchical [12] CPN model representing common topology structure of ICS architecture can be defined. This generic CPN model is able to represent any type of ICS topology. This hierarchical CPN model is the CPN model holder during the automatic model generation and will be called "CPN holder" (Fig. 2).

The CPN holder model is based on defining generic links between several component families. A family gathers generic components involved in ISC architecture (SCADA or PLC for example). A family may embed several different components through an instantiation process. At last, a family instantiates some generic CPN models of elementary components. Components having similar behavior and operating mode are modeled using a unique CPN generic model. Those CPN models are customizable by using a set of parameters defining their specific features (for example, two PLC of the same family may be able to process different amounts of requests per scan). Each component is modeled with a generic interface (Tx places for outgoing messages from the components and Rx places for incoming messages) respectively connected to an output buffer and an input buffer (Fig. 3). Also the specific behavior of a component has to be represented inside the substitution transition "Functional architecture component" in the Fig. 3.

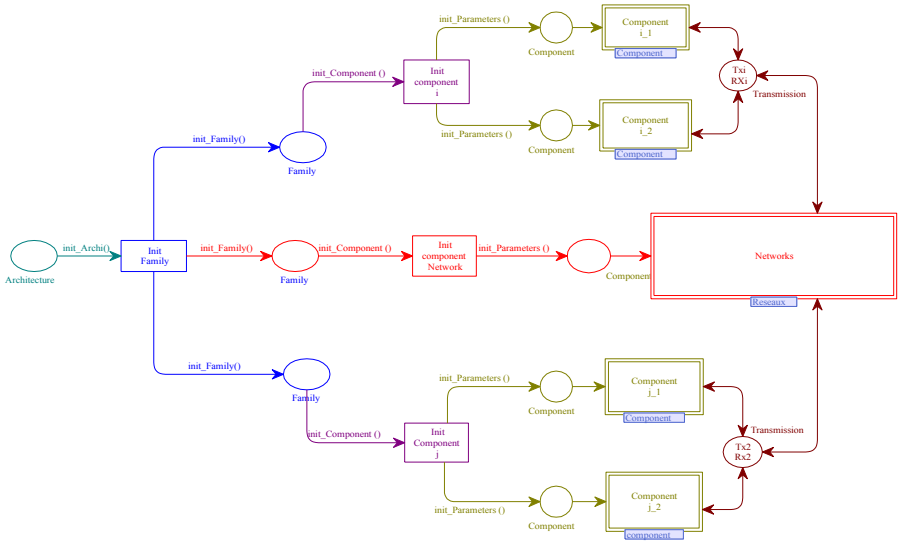

Fig. 2 CPN holder

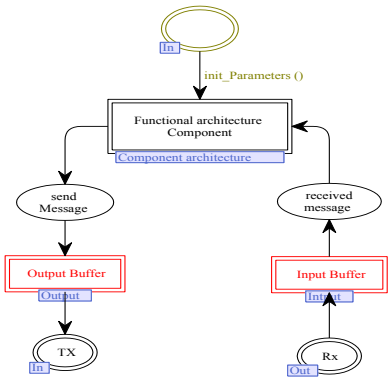

Fig. 3 Mandatory component CPN structure

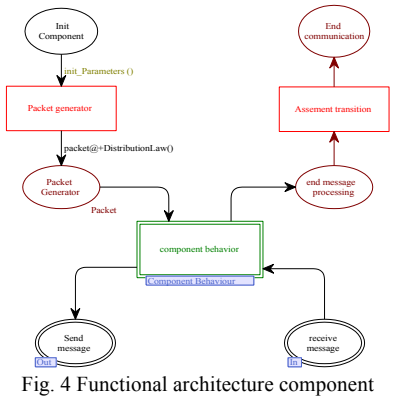

Fig. 4 Functional architecture component

The Fig. 4 provides more details about component internal behavior. One more time, the internal structure is generic: the component dotted with a packet generator (that represent the fact that any component can be the spontaneous emitting source of a message) and with an observer transition ("*assessment transition*") holding all monitors [12] related to performances of this component family. Note that the ML function that is triggered when firing the "*packet generator*" transition is specific for each component. Some components are designed to periodically sent messages (supported by a timed deterministic function associated to the output arc of "*packet generator*" transition), while other ones may randomly sent messages. For example SCADA sent messages described by a Poisson law when it corresponds to a user request and by a uniform distribution when it corresponds to a refresh request. At last, the substitution transition "*component behavior*" contains the specific internal behavior of each component.

The tokens that are involved in CPN holder, family and components models represent a message, which is exchanged across the network between some of those components. This "communication token" must have at least two colors set which are the source of the token and the destination of the token. The value of source color set will be the ID of the component generating the token. The values of the destination will be the ID of the component receiving the token.

Parameterization of the model and instantiation of components used in the families, as well as families used in the ICS architecture, are done through the token and color set declaration [12]. The basic idea is to define a generic and hierarchic CPN structure using substitution transition, which is customizable for a given ICS thanks to the initial marking. Parameterization and instantiation process the following steps.

B. *Parameterization and instantiation*

Automatic CPN model generation is done through three mains steps.

Component (for example *Component i_1* in the Fig. 2) may contain several devices of the same kind; the token within the place "*component*" embeds the identification of devices that are present in a given ICS architecture and their parameters. If devices have different behaviors or different operating modes, different CPN models are used for their description (this is the case in Fig. for *component i_1* and *i_2*).

Family is a generic class of components (SCADA, PLC, I/O devices, Network); the token within the place "*family*" in the Fig. 2 contains the instantiation parameters of the family.

The Global ICS architecture may (or not) involve the whole identified families; the token within the place "*Architecture*" contains information for instantiating these families to give rise to the ICS architecture.

The definition of the token colors within these three parameterization and instantiation places (*component*, *family*, *architecture*) is based on the definition of ML function [13].

The initial marking within the place "start" must be automatically with the XML description of the ICS architecture. Based on this initial marking, ML functions *init_archi()*, *init_family()* and *init_component()* successively instantiate the family and components involved in the ICS architecture. For example, a token will be generated to instantiate a family model within a given ICS architecture only if this family is involved in it. The same rationale is applied for instantiating components that are used by a family instance. Note that we assume that the network family will be always instantiated. Once these instantiations have been processed, the function the *init_parameters()* function parameterizes instantiated components based on their specifications (specific internal features such as periodic time scan, parameters of the probability distribution, …).

C. *Automatic generation of CPN model*

Based on the rationales for modeling, instantiation and parameterization that are presented in the two previous sections, the algorithm of the Fig. 5 summarizes and schedules the different automatized tasks that has to be performed for transforming an informal description of the ICS architecture into a CPN model as required for its performance assessment.

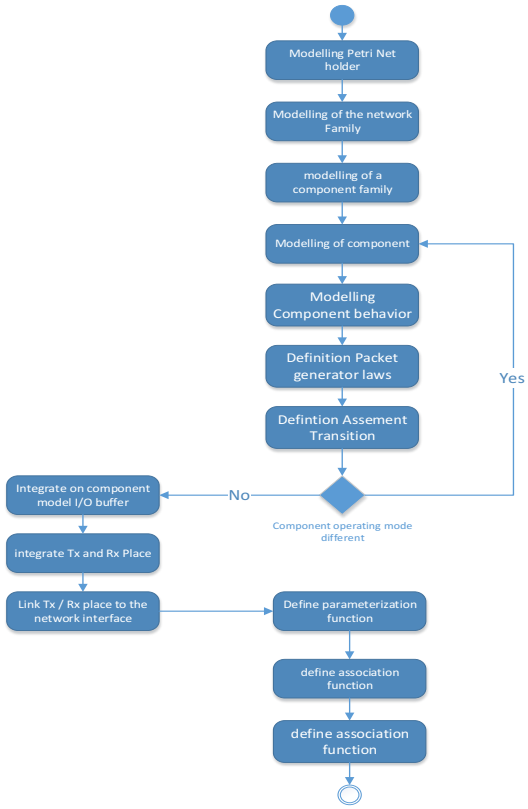

Fig. 5 Algorithms of automatic model generation

Thereby with these colored tokens, instantiation and parameterization ML function, the CPN model of the architecture is generated automatically without manually defining the model. Only parameterization functions have to be written regardless the architecture to define. Then by using

monitors [12] on defined transition, performance can be assessed.

IV. Application

A. Case study

In this section a small architecture is used in order to illustrate the automatic CPN model generation of ICS architecture. The goal of this section is to show how easy the automatic reconstitution is and may be time saving. Also how the automatic reconstitution methodology provides a relevant solution to our problematic defined in the section 1.

The Fig. 6 shows a simple architecture composed with 2 clients, 1 PLC, and 3 devices. The clients send periodically requests to the PLC. The PLC will process the requests from the client, scan the devices and retrieve information from them and then update their memories during its cycle time.

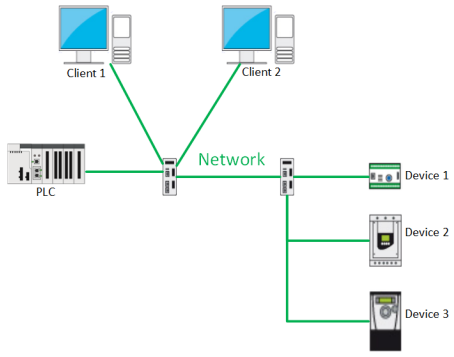

Fig. 6. Simple architecture

B. Automatic generation of the CPN model

The "CPN holder" model of Schneider ICS architecture is given in Fig. 7 taking into account that four families have been defined for Schneider architecture: client (user interface or SCADA), PLC, I/O devices and networks.

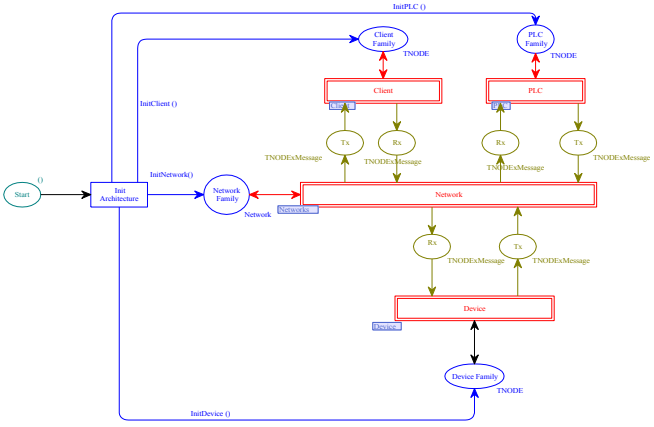

Fig. 7. Automatic model definition of the simple architecture

To be able to generate the CPN model of the simple architecture, color set must be declared in order to configure the model. The first declarations are for creating Ids and also the communication token.

```
(*Declaration of component ID*)
Colset ID_component = INT
Var ID: ID_component;
(*Declaration of communication token*)
Colset Source = INT ;
Colset Destination = INT ;
Var src : Source ;
Var dst : Destination ;
Colset TokenCommunication = product Source *
Destination;
```

Once the colors set related to the model structure are done, components must be configured based on their parameters and associated with their families. The client has one parameter which is the sending period. The PLC also has one parameter which is the application cycle time. Finally devices have as parameter the scanning rate. In addition to these parameters, each component must have an *IDcomponent* color set included in their configuration. These IDs allow identifying them.

```
(*Declaration of client *)
Colset RequestPeriod = INT;
Var period: RequestPeriod;
Colset Client = record ID:IDcomponent *
period:RequestPeriod;
(*Declaration of PLC *)
Colset PLCcycleTime = INT;
Var cycle: PLCcycleTime;
Colset PLC = record ID:IDcomponent *
cycle:PLCcycleTime;
(*Declaration of Devices *)
Colset ScanTime = INT;
Var scan: PLCcycleTime;
Colset Device = record ID:IDcomponent *
scan:ScanTime;
(*association of component to their families*)
Colset config_familly = union Client:InitClient +
PLC:InitPLC + Device:InitDevice
```

The definition of previous color set allows the generic configuration of the model. Thereby regardless the architecture defined these colors set will be used in order to generate automatically the relevant CPN model of the architecture. These colors set configuration and CPN model definition are done once by the expert of CPN modeling.

Once the CPN model and color set declaration are done by the CPN modeling expert, parameterization of the desired architecture is achieved by simply define these ML functions. The code below shows the parameterization of the simple architecture in the Fig. 6.

```
(* there are two clients in the architecture
sending request in a different period*)
Val client = ref [Client ({ID_component = 1, period =
50}); Client ({ID_component = 2, period = 100})]
Fun InitClient () = (!client)
(* there is one PLC in the architecture with a
cycle*)
Val PLC = ref [PLC ({ID_component =3,cycle = 70})
Fun InitPLC() =( !PLC)
(* there are three device in the architecture with a
different scan time*)
Val device = ref [Device ({ID_component =4, Scan =50}),
Device ({ID_component t =5, Scan = 100})}), Device
({ID_component t =6, Scan = 150}) ]
Fun InitDevice () = (!Device)
```

Once the CPN model is parameterized with the relevant function, tokens corresponding to the components architecture

are created when the transition init architecture is bounded. Created tokens will represent each component on the architecture and will have as color set values declared on the parameterization function. Thereby the model of the architecture is generated and the simulation can be started.

C. Performance assessment

Main temporal performance to be assessed is the end-to-end delay required for a message to go from a sender device to a receiver device. This performance is monitored thanks to:

- A time stamp that is assigned to the tokens leaving the output buffer of a device; this requires the definition of an additional color for performance purposes: product `Source*destination*timestamp`).
- A monitor [12] is triggered when the transitions representing the reception interface of a component are fired. This monitor will compute the difference time between the current time of the simulation and the time stamp of the emitted token. Each device has its own monitors in the same transition before "input buffer" place. The link between emitted tokens and received tokens is made thanks to the IDs of the sender and receiver device and identification of the message.

The Table 1 shows the result of two simulations. It shows the end-to-end delay for a message sent by a client (SCADA for example) to different I/O devices (Device 1, 2 and 3) using a PLC as intermediate station. The difference between the obtained results is mainly justified by the difference scan time of the three devices.

	Device 1			Device 2			Device 3		
	min	avrg.	max	min	avrg.	max	min	avrg.	max
Simulation 1	53	70.4	123	103	110.4	173	153	150.4	223
Simulation 2	55	70.8	122	104	111.6	175	152	149.2	221

Table 1. Result of performance assessment small architecture (time in ms)

A similar approach can be applied to the different temporal performances to be assessed for validating a given ICS architecture.

V. Conclusion

This paper has demonstrated firstly that CPN is a relevant choice for modeling and assessing the performance of ICS architecture. From the problematic imposed by the pre-sales context, the paper shows the relevant solution of generate automatically a CPN model. This automatic generation has 3 mains advantages; firstly the CPN model holder and the configuration are done only once. Secondly a user without experience on CPN modeling can easily with parameterization generate CPN models. Finally this allows testing wide range architecture without spending time on the modeling. Thereby this automatic model generation is a huge step forward for industrial companies to use CPN as model definition and simulation. However the methodology is facing 3 mains limitations which are:

- The size of the parameterization and instantiation token which can be huge based on the complexity of the architecture. This size increase the simulation time the next work will focus on reducing this size of the token by defining more efficient ML functions.
- The network family has been modeled at this stage of the work as a random delay which is not the real behavior. The next work will focus on modeling the Ethernet and other fieldbuses behavior and network components.
- The simulation provides performance as an average value. However for some architecture the maximum value is required. An investigation will be made for combining the simulation approach with worst case approaches.

References

[1] J. Jasperneite and P. Neumann, "Performance evaluation of switched Ethernet in real-time applications," in 4th IFAC, 2001.

[2] R. Winkler, "Uncertainty in probabilistic risk assessment," Reliability Engineering & System Safety, vol. 54, pp. 127-132, 1996.

[3] L. Seno, S. Vitturi and C. Zunino, "Real Time Ethernet Networks Evaluation Using Performance Indicators," in Emerging Technologies & Factory Automation, 2009.

[4] R. Zwick and T. Walisten, "Combining stochastic uncertainty and linguistic inexactness: theory and experimental evaluation of four fuzzy probability models," International Journal of Man-Machine Studies, vol. 30, no. 1, pp. 69-111, 1989.

[5] J.-P. Georges, T. Divoux and E. Rondeau, "Network calculus: application to switched real-time networking," in Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools, 2011.

[6] R. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation.," in IEEE Trans. Information Theory, 1991.

[7] A. Boussad, A. Saıd and J.-J. Lesage, "Genetic algorithms for delays evaluation in networked automation systems," Engineering Applications of Artificial Intelligence, vol. 24, pp. 485-490, 2010.

[8] G. Marsal, "Performance analisys of industrial ethernet networks by means Timed Model-Cheking," in 12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM, Saint-Etienne (France), 2006.

[9] S. Jackman, "Estimation and Inference via Bayesian Simulation: An Introduction to Markov Chain Monte," American Journal of Political Science, pp. 375-404, 2000.

[10] B. Brahimi, E. Rondeau and C. Aubrun, "Integrated ap-proach based on high level Petri Nets for evaluating Networked Control Systems," 16th Mediterranean Conference on Control and Automation MED'08, Jun 2008.

[11] K. Jensen, "An Introduction to the Theoretical Aspects of Coloured Petri Nets," Springer, Berlin Heidelberg, 1994.

[12] K. Jensen and L. Kristensen, Coloured Petri Nets modeling and validation of concurrent Systems, Berlin Heidelberg: Springer, 2009.

[13] R. Milner, The definition of standard ML: revised., Cambridge, MA.: MIT press., 1997.

[14] J. Jasperneite and P. Neumann, "Switched Ethernet for Factory Communication," Emerging Technologies and Factory Automation, pp. 205-212, 2001.