Practical Use of Coloured Petri Nets for the Design and Performance Assessment of Distributed Automation Architectures

Moulaye Ndiaye[1,2,3], Jean-François Pétin[2,3], Jean-Philippe Georges[2,3], and Jacques Camerini[1]

[1] Schneider Electric - Process automation, Carros, France
{moulaye.ndiaye, jacques.camerini}@schneider-electric.com
[2] Université de Lorraine, CRAN, UMR 7039, Vandœuvre-lès-Nancy, France
[3] CNRS, CRAN, UMR 7039, Vandœuvre-lès-Nancy, France
{moulaye.ndiaye, jean-francois.petin,
jean-philippe.georges}@univ-lorraine.fr

Abstract. Distributed automation system (DAS) architecture gathers all the hardware and software automation components in order to realize control and monitoring tasks for factory process. To submit and commercially secure an architecture offer, system integrators must assess the performances of a wide range of candidate architectures, which enforce functional requirements. Among many available methodologies, Coloured Petri Nets (CPN) have already proven their capabilities to assess DAS architecture performances. However, performance assessment using CPN involves a manual definition of the required models which is time-consumer, when a huge amount of different architectures must be considered. This paper introduces a practical CPN-based methodology to automatically generate the relevant architecture CPN model by instantiation and parameterization of generic CPN components model. It allows easily defining, assessing and validating the architecture. This study is supported by Schneider Electric Company and illustrated with process automation reference architectures.

Keywords: Coloured Petri Nets, performance assessment, distributed control architecture, model synthesis.

1 Introduction

A Distributed Automation System (DAS) is composed of several processing devices such as: Supervisory Control and Data Acquisition (SCADA), Programmable Logic Controllers (PLC) or I/O devices (actuators, sensors, transmitters) implementing all required functions for controlling and monitoring an industrial system. During the pre-sales phase of an automation project, system integrator (SI) defines DAS proposals that fulfil specifications contained in customer's call for tender. This pre-sales phase is characterized by low information about the

project, short time range to submit an offer, and limited financial resources and manpower.

Available information provided by the customers is generally limited to the piping and instrumentation diagram (P&ID) which contain the list of actuators, transmitters and sensors of the process system to be controlled and the process operating control. This P&ID diagram is associated with the specifications of the minimal DAS performances in terms of:

- temporal performances gathering all response time of the architecture and defined by the time from end-to-end delays, latencies and freshness (from the occurrence of an input and the activation of an output);
- PLC scan time defining the amount of time takes by PLC's CPU to perform its entire automation tasks based on the process and external inputs;
- load performance, defining the load rate of components having limited resources and sharing it with other components in order to operate control/command tasks.

With these limited information, SI in charge of defining and implementing DAS architecture selects from automation component manufacturer hardware and software that fulfil the I/O and P&ID requirements. A wide range of DAS architectures may answer to the functional requirements but only some of them fulfil the required performances. To achieve performances assessment of candidate DAS, SI are currently using 3 solutions. The first one consists in designing the architecture on the basis of manufacturer's reference architectures that provide some guarantee on the theoretical behaviour and performances. The main advantage of this solution is to reduce the time and the resources for submitting a commercial offer. However, once the architecture is deployed, some gaps may be noticed between expected and implemented performance that lead to architecture modifications, cost and penalties for the SI. The second solution is to oversize architecture through 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 these solutions inspire Schneider Electric to develop a new solution able to assess the performance of the DAS 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 several and various ICS architectures without engaging huge time, manpower and financial resources.

Section 2 introduces the main industrial and scientific requirements of the expected performance assessment software tool. Section 3 presents the related works. Section 4 details our contribution enabling automatic generation of Coloured Petri Nets models to support a fast assessment of several potential architec-

tures. Section 5 provides a case study based of Schneider Electric DAS architectures. Finally, section 6 concludes and gives direction for future works.

2 Requirements

2.1 Industrial requirements

The main requirement about the expected software tool for DAS performance assessment is illustrated on the Fig. 1. The Input of the tool is the description of the DAS architecture through domain specific language (DSL) [16] that will be provided by the System Integrators. It contains the description of commercial components which are involved in DAS architecture, the material links between these components and the main information flows that are exchanged between them. This description also defines DAS performance that must be assessed. From it, the tool must provide performance assessment:

- requires building a formal model of the DAS architecture representing the behaviour of the DAS architecture;
- compute DAS performances from the DAS formal model. If the assessed performances are not compliant with the specified ones, the tool must allow a quick modification of the architecture from the DSL.

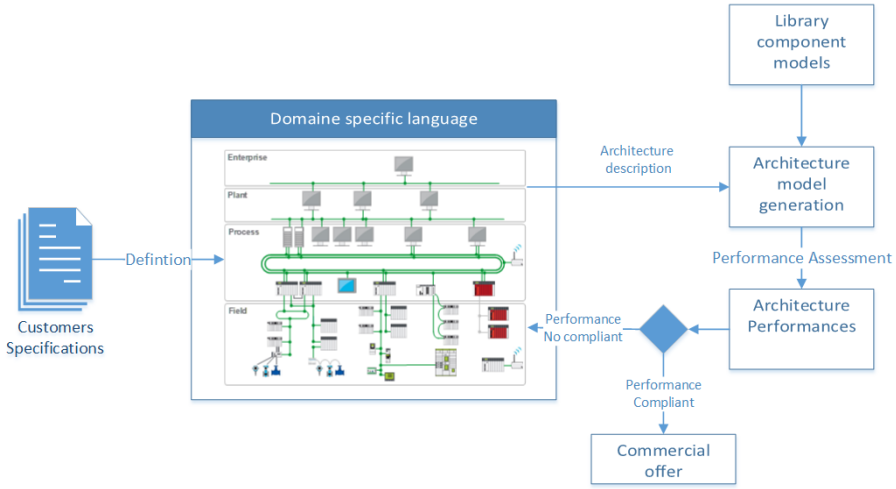


Fig. 1. DAS performance assessment tool

Taking into account the need to assess a wide range of potential architectures, those two steps (modelling and assessment) must be, as most as possible,

automatic to be compliant with the limited resources and time pressure during the pre-sales stage. The opportunity to develop a library of DAS component behaviour model, based on existing commercial components, is a significant step towards to the automatic DAS model generation. In other words, DAS behaviour model is expected to be automatically generated from the topology of the DAS architecture and by instantiating it library of component behaviour.

2.2 Scientific requirements

The main challenge encountered in the pre-sales context is related to the uncertainty modelling. This uncertainty covers several aspects of the timed behaviour of the DAS architecture [15][19]:

- the features of the DAS architecture are not precisely known: the behaviour of the component and the use context of the control system can only be approximately estimated by the SI (for instance, the processing time of PLC depends on the execution of embedded software, which can only be approximated at this stage, through a uniform distribution for example);
- the events occurred on the DAS architecture (messages emitted by sensors and actuators, control actions sent by the human operators, ...) have to be considered as random events in order to model several operating modes; these random events are spatial (the DAS components exchanging information are not precisely known) and temporal (the occurrence time when a device initiate a communication is not known neither).

Consequently, the formalism to be used for performance assessment must enable the characterization of timed behaviour, including random input parameters with probability laws distribution, some of them may be non-exponential (uniform distribution, Poisson, Weibull, ...).

Due to the complexity of industrial process, DAS architectures are structured into hierarchical levels. Each level gathers components based on the services they provide, the type, the size and the frequency of exchanged information [8]. These information are exchanged within each level and also between different levels through communication networks. Moreover, components providing same services can operate with slight behavioural differences when providing them. Those points lead to additional requirements:

- 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;
- formalism has to support the modularity, re-usability and parametrization of the models.

The scientific problem addressed by this paper focus on the generation of DAS model using a formalism that support the previous requirements in order to quickly provide models and performance assessments for a wide range of DAS architectures.

3 Related works

Performance assessments of DAS architecture have been studied during these past years. Many methodologies have been developed through two main families: analytic evaluation and simulation.

Analytical evaluation uses formal calculus to compute the end-to-end delay for DAS 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, real time calculus, trajectory approach or (max+) algebra, which are determinist theories of queuing systems [4][5][1] or the model-checking which is based on the formal verification of time communication between components [3][18]. Usually, these theories assume deterministic knowledge of the traffic exchanges. Other approaches rely on probabilistic models, such as Markov Chain theory [7], providing analytically an estimation of the response time. If system models are homogeneous to a Markov or Semi-Markov processes (mainly while using only exponential distributions). This limitation, with regard to the probabilistic features of the DAS requirements, makes this method unsuited for this context.

Simulation consists of an event-driven or real-time execution of the model. Modelling a DAS architecture basically relies on models from the Discrete Event System (DES) theory (timed automata, Petri Nets, DEVS, ...). In case of probabilistic model, several traces must be executed to compute statistical information about the performance. It is generally done using Monte-Carlo simulation using probabilistic state-based models.

Among many methodologies that have been studied, Petri Nets have already proven their efficiencies on modelling and assessing DAS performance. In particular, Coloured Petri Nets (CPN) [9][10] have proven their efficiencies of modelling and assessing performance of DAS. In [12], the models focus on DAS control devices whereas network performances have been retrieved from experimental benchmark studies and in [2], the modelling is extended to specific network devices (Ethernet switches).

CPN is a discrete-event modelling language combining the capabilities of Petri nets with a high-level programming language. Main differences with ordinary PN are:

- colours identifying and characterizing a token with different data types (e.g. Boolean, integer, string, or more complex data structure);
- hierarchical concept that promotes the modelling of complex CPN by combining several small CPNs; it overcomes the lack of compositionality, that is one of the main critiques raised against Petri net models;
- CPN may be extended to time concept (deterministic or stochastic delay).

Time is modelled using a time value, also called a time stamp, adding a tag to the tokens in addition to their colours. When a transition is enabled, it is fired and changes the time stamps of tokens which are deposited in its output places. In these places, the tokens remain frozen and can not be used for enabling other

transitions until the current model time (as given by the global clock) is smaller than their time stamps. As soon as the time stamp of the tokens is less than or equal to the current time model, these tokens can enable other transitions which are instantly fired. In other words, this behaviour matches the formalised theoretical behaviour of P-timed Petri net operating at its maximum speed transition. Moreover, timed transitions may be fixed (deterministic) or stochastic, i.e. which fire after a random enabling time. At last, CPN may combine immediate transition (without time constraints) and stochastic transitions. Such behaviour is covered by a particular class of Generalized Stochastic Petri Nets (GSPN) defined in [11]. Regarding the immediate transitions, CPN behaviour is the same than in GSPN, but unfortunately, the behaviour of the stochastic transitions differs to the firing of stochastic transition that is in competing. A mechanism that force a sojourn time in the input place of a stochastic transition has been proposed by [14] to ensure a behaviour that is compliant with formal GSPN with enabled memory policy for firing transitions. Main benefit of using CPN instead of GSPN in our case is that these distributions laws are not limited to exponential ones, the drawback is that, our models will be limited to the use Monte-Carlo simulation.

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 [10] for modelling, simulation and performance assessment. Thereby functions described below, use the CPN tool formalism through the Standard ML programming language [13]. However, it must be noticed that, CPN definition is independent of the concrete inscription language. It involves that any other language than CPN ML can be used as long as the selected languages includes data type, variable declaration, expression definition, value bounding and functions definition is available.

Nevertheless, in the several related works about performance assessment using CPN, the models have been specifically defined manually for a given DAS architecture. The objective of this paper is to define a modelling framework that enable automatic generation of CPN models for performance assessment in the pre-sales context.

4 Proposal

4.1 Generic DAS architecture modelling

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

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

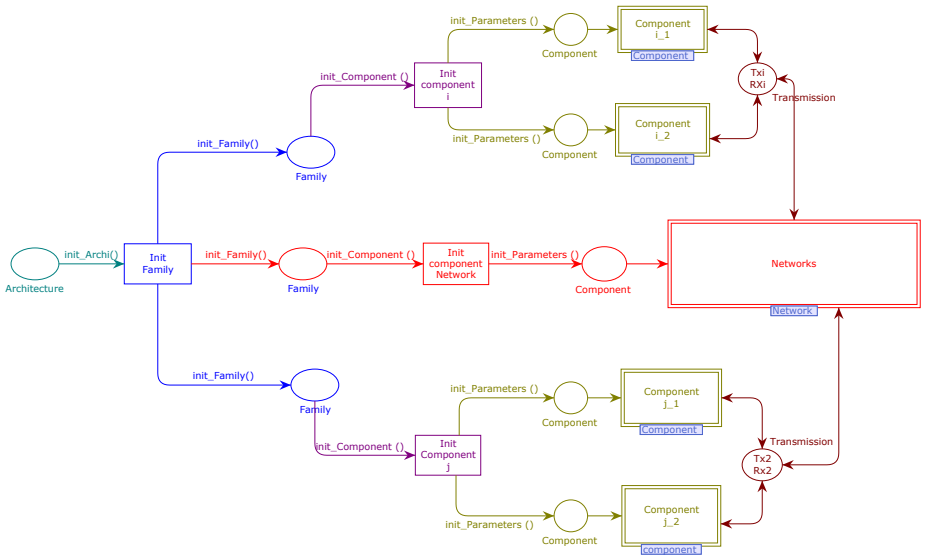
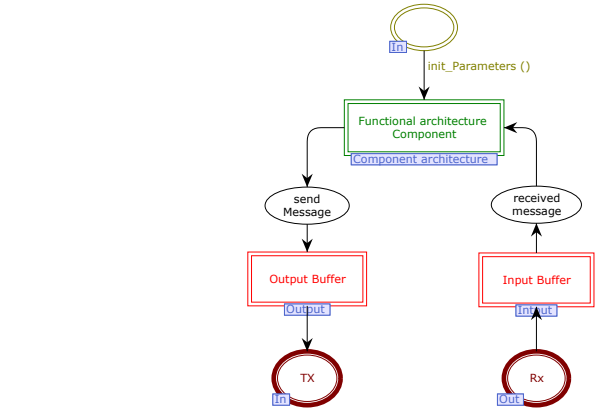


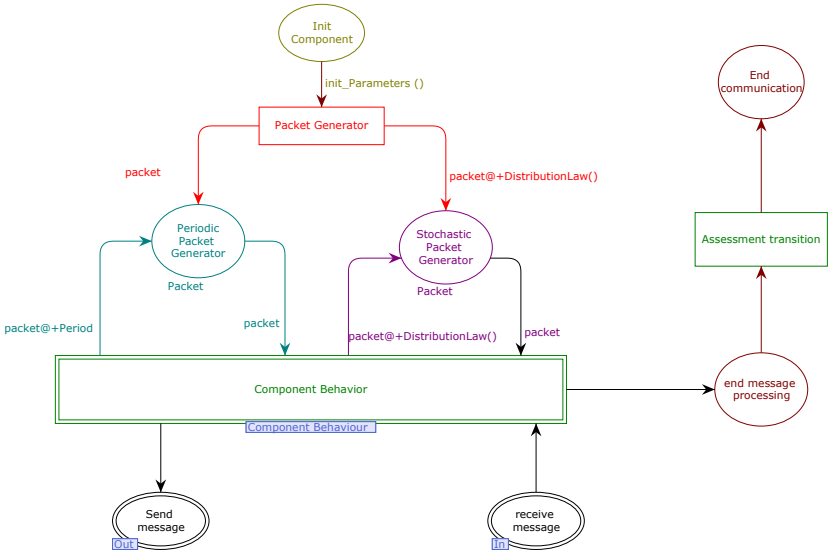
Fig. 2. CPN Holder

The *CPN holder* model is based on defining generic links between several component families. A family gathers generic components involved in DAS 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 behaviour and operating mode are modelled 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 modelled 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(a)). Also the specific behaviour of a component has to be represented inside the substitution transition *Functional architecture component* in the Fig. 3(a).

The Fig. 3(b) provides more details about component internal behaviour. One more time, the internal structure is generic: the component is dotted with a packet generator (representing the fact that any components can be spontaneously the emitting source of a message) and with an observer transition



(a) Generic interface



(b) Functional structure

Fig. 3. CPN component generic model

(*assessment transition*) holding all monitors 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 and depends on the way the component is initialized. The components may sent periodic messages (modelled by deterministic durations using a timed function associated to an output arc of *packet generator* transition), or non periodic messages (modelled as a random emission thanks to a distribution law associated to another output arc of *packet generator* transition). For example, SCADA sends messages described by a Pois-

son law when it corresponds to user request and by a uniform distribution when it corresponds to a refresh request. At last, the substitution transition "component behaviour" contains the specific internal behaviour of each component.

Generic components on the *CPN holder* are connected to a sub transition called *Network*. Different protocols might be used in DAS. Since a large set of modern protocols is based on switched Ethernet (especially those used in Schneider Electric solutions), we consider here only the modelling of low layers (Ethernet). The proposed work could be then extended by taking care to application layer protocols like Modbus/TCP, Profinet or Ethernet/IP. It would require to model parameters like frame format and communication model (like slipstreaming effect or master/slaves) into the generic components. An exhaustive model would have to take into account the different network protocol states and it would lead hence for protocol like TCP to a huge amount of places and arcs in each component. In this paper, the related sub transition represents an Ethernet switch with multiple ports for each network. It will enabled to tackle the delay and packet loss that may happened on such device. As shown on the Fig. 4, the generic model of this switch is composed with a port for each generic component model. This port connects generic model of a component through the *TX* and *RX* place and has a unique ID. This ID is static value fixed while defining the *CPN holder* through generic component families. Add to these ports, there is a buffer storing and forwarding the tokens between ports based on the arrival order. Finally, there is the switching table place holding the architecture switching table that must be parametrized.

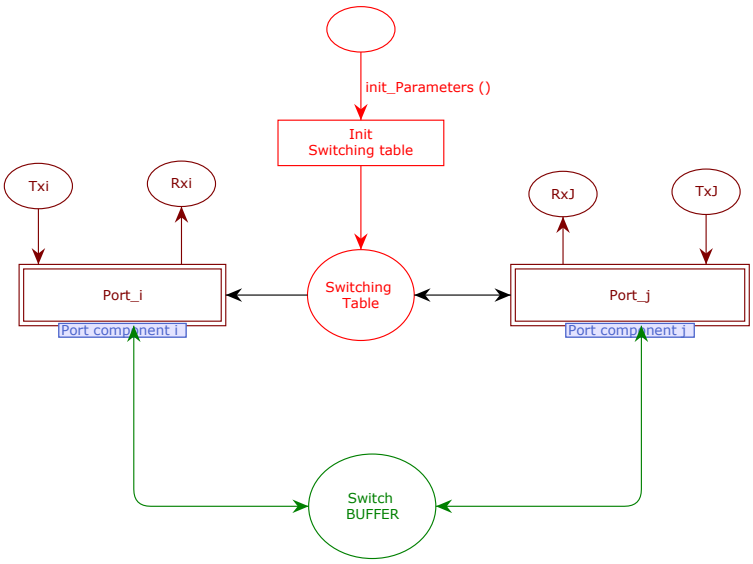


Fig. 4. Swicth model

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 colours set which are the source of the token and the destination of the token. The value of source colour 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. Parametrization of the model and instantiation of components used in the families, as well as families used in the DAS architecture, are done through the token and colour set declaration. The basic idea is to define a generic and hierarchic CPN structure using substitution transition, which is customizable for a given DAS thanks to the initial marking.

4.2 Parametrization 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 components that are present in a given DAS architecture and their parameters. If components have different behaviours or different operating modes, different CPN models are used for their description (this is the case in Fig. 2 for *component i_1* and *component 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. For example, tokens in the place *switching table* are coloured by the product of two colours set: the unique ID of all instantiated components of the Architecture representing for the switching table the MAC address of the component and the unique ID of the port on which the component is physically connected. Thereby, there are as many as switching table tokens than instantiated automation and networks (switches) components.

The global DAS 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 DAS architecture.

The definition of the tokens colours within these three parametrizations and instantiations places (component, family, architecture) is based on the definition of ML functions. The initial marking within the place *start* must be automatically generated from a Standard Generalized Markup Language (SGML) [6] describing the DAS architecture.

Colour sets are created based on the parameters of the components. That means each DAS architecture components has specific parameters, and for each parameters a colour set must be declared. The colour set defining a component is done by associating all parameters colour set. With CPN ML, this association can be done with the declaration type *record*. Once all colour set related to parameters for each component are defined, then associated to the colour set of the component, the configuration colour set can be created. This configuration colour set is a union (CPN ML *union*) of all components colour set in order to create an entity representing a CPN model in terms of colours set of a DAS architecture.

To illustrate these colours sets declaration, let's consider four component families (Client, SCADA, PLC and I/O Device). Let's consider Client has three parameters which are $Cp1$, $Cp2$ and $Cp3$. The two first parameters represent an integer values and the last one represent a string value. SCADA has two parameters both are integer type and called $Sp1$, $Sp2$. PLC has one integer parameter called $PLCp1$. Finally, I/O devices have one integer parameter called $Dp1$. First of all, colour set corresponding to the parameter type of each component must be declared. In CPN ML the declaration is done as follow:

Component parametrization
colset Cp1 = INT; ***colset*** Cp2 = INT ; ***colset*** Cp3 = STRING;
colset Sp1 = INT; ***colset*** Sp2 = INT ;
colset PLCp1 = INT; ***colset*** Dp1 = STRING;

Then the colour set parameters are associated to a component in order to create the component family colour set. For that a variable holding the value of the colour set parameters is associated to the colour set previously defined.

Family parametrization
colset Client = ***record*** cp1:Cp1 * cp2:Cp2 * cp3:Cp3;
colset SCADA = ***record*** sp1:Sp1 * sp2:Sp2;
colset PLC = ***record*** plcp1 : PLCp1; ***colset*** Device = ***record*** dp1 : Dp1;

Finally, the architecture colour set is defined by combining all component colour set. This is done by association the component colour set to a variable holding the component instantiation and configuration.

Architecture parametrization
Colset Architecture = ***union*** Client:ClientFamilly + Scada:ScadaFamilly + PLC:ControllerFamilly + Device:DeviceFamilly

Based on this initial marking, the ML functions $init\_archi()$, $init\_family()$ and $init\_component()$ successively instantiate the family and components involved in the DAS architecture. For example, a token will be generated to instantiate a family model within a given DAS 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 $init\_parameters()$ function will parametrize the instantiated components based on their specifications (specific internal features such as periodic time scan, parameters of the probability distribution, ...).

4.3 Automatic generation of CPN model

Based on the rationales for modelling, instantiation and parametrization that are presented in the two previous sections, the Fig. 5 provides an overview of the DAS model generation. It summarizes and schedules the different tasks to

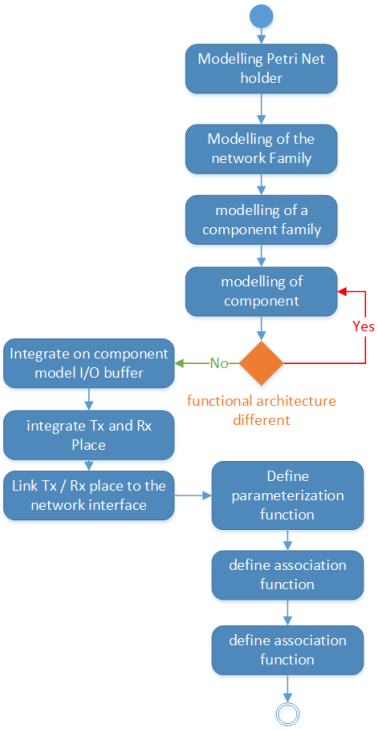


Fig. 5. Overview of DAS CPN model generation

perform the transformation of an informal description of DAS architecture into a CPN model as required for its performance assessment.

Thereby with these coloured tokens, instantiation and parametrization ML function, the CPN model of the architecture is generated automatically without manually defining the desired model. Only parametrization functions have to be written regardless the architecture to define. Then by using monitors on defined transition, performance can be assessed.

5 Application

This section illustrates the automatic model generation through instantiation and parametrization tokens. It shows how fast and easy is the generation of CPN model of any DAS architecture regardless it size, complexity or topology. However, we warn the reader about the fact that the CPN DAS model generation is based on the DAS unformal description but also on the *CPN holder* and a library of CPN component models that have been previously defined by a CPN expert and that are specific for each DAS supplier company.

5.1 Case study

Two case studies have been selected to illustrate the generation of DAS models.

The first one is the (Fig. 6) representing an example of small DAS architecture, using Schneider Electric component for water treatment in small size cities. This architecture is based on a linear full-segmented network where all automation components are connected to a switch. The three hierarchical levels (operation, control and device) of DAS architecture are here interconnected by Ethernet-based protocols.

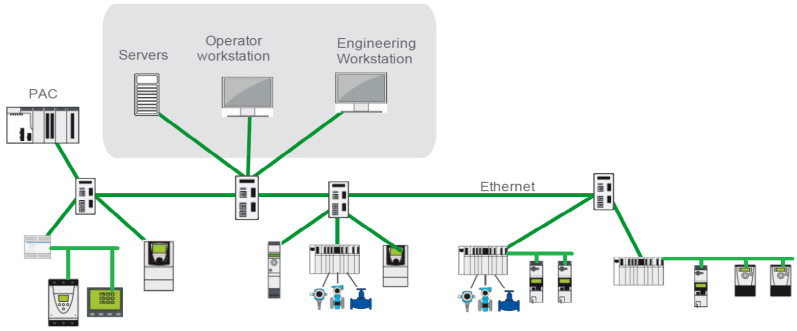


Fig. 6. Small DAS architecture dedicated for water treatment plan

The second DAS architecture (Fig. 7) represents a mid-size architecture for cement plant. This architecture is separated into three communication layers: the operation network, the control network and the device network. All these networks are based on the Ethernet protocol. Each communication layers corresponds to a DAS structure level.

5.2 Automatic generation of the CPN model

In order to illustrate the automatic model generation, a *CPN holder* has been defined (Fig. 8) to meet the structure of any Schneider DAS architecture. This model shows 4 families of components. The client family gathering all the operator and engineering workstation, batch server etc. The SCADA family representing the monitoring and acquisition system PLC & PAC family gathering all controllers and Devices family gathering end devices for task execution.

All these families are connected to the network family representing the model of a network switch. Inside each family, components are described by the CPN generic structure of Fig. 3. Component interface consists in a monitor transition and two places *RX* and *TX* for the communication as well as a token generator for initiating a communication. Specific behaviours of the components are stored in a library that has previously been built by component and CPN experts. For confidentiality matters, the CPN model of Schneider components are not shown.

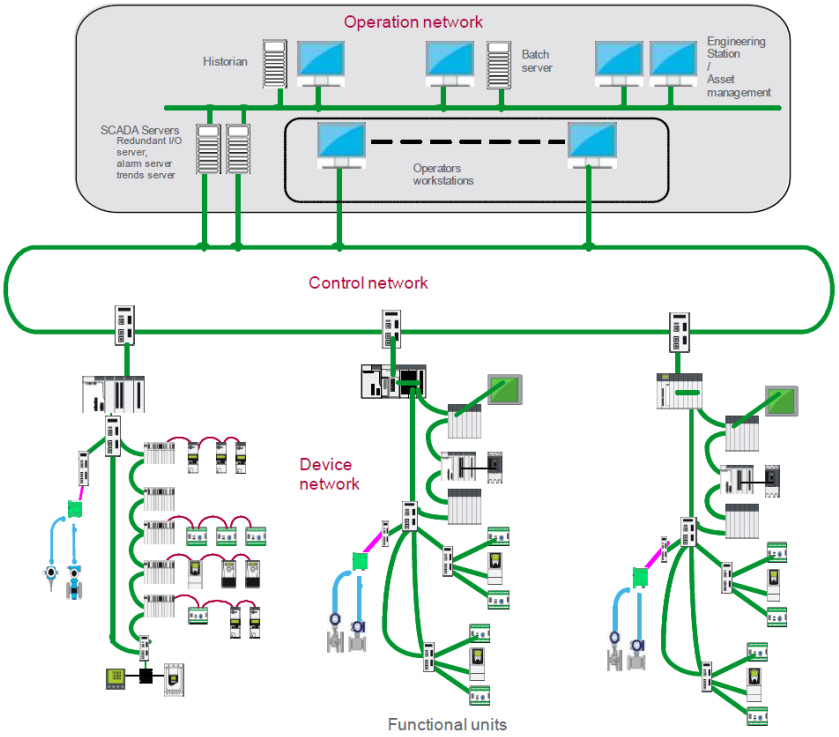


Fig. 7. Mid-size DAS architecture dedicated for cement plant

The *CPN holder* model is manually defined once by the expert on Petri nets modelling as well as the CPN component library. Performance model are automatically generated from the CPN holder and component library thanks to initializing tokens. This initialization is done by firstly generating an SGML file (through an XML file for instance) from the informal DAS description using DSL, including the parameters specification (number of components, input parameters etc..). Secondly, this SGML file is processed using a parser algorithm [17] in order to extract values. Finally, the extracted values are assigned as colour set to the ML functions of each component family (refer to section 4.2).

To illustrate the initialization process for architectures of Fig. 6 and Fig. 7, the client, SCADA and PLC value declaration is presented. Same rationale is used for the I/O devices components but their large number does not allow a presentation of parametrization tokens within this paper (Fig. 6 involves 2 client workstations, 1 PLC and 14 devices while Fig. 7 involves 6 client workstations, 3 PLC, 1 SCADA and 40 devices.). Let's consider $Val_{integer}$ as an integer value and Val_{string} as a string value. Considering that all values of these variables can be automatically obtained by parsing the XML file that describes the DAS architecture, parametrization is given by Table 1.

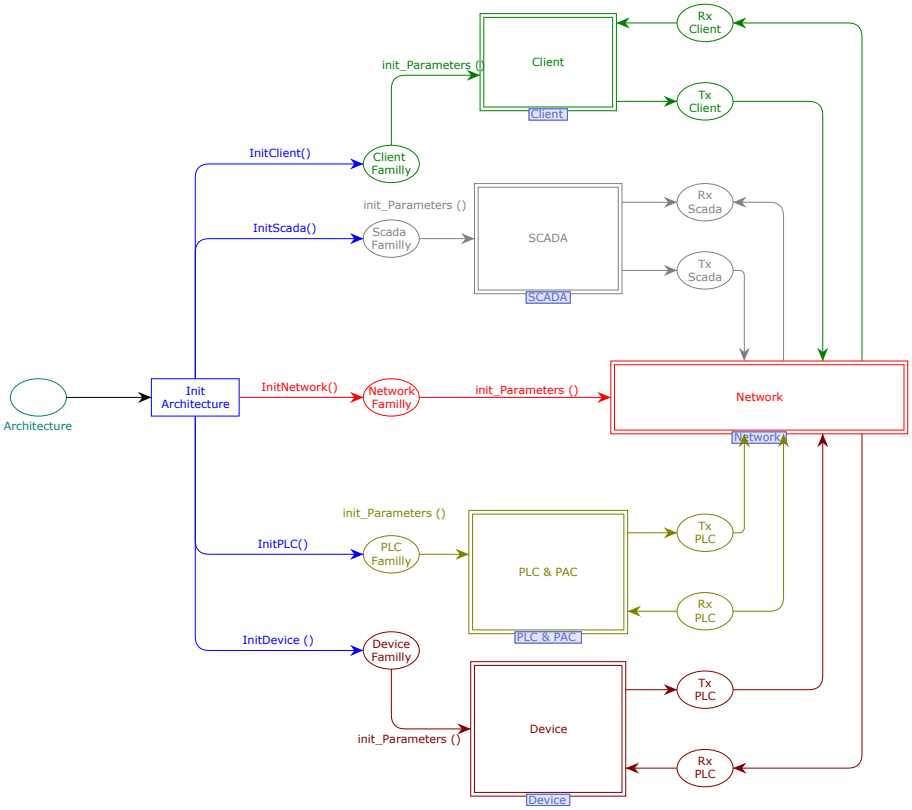


Fig. 8. Schneider CPN holder

When the transition *init architecture* is fired, the functions are initialized based on the values previously assigned, and then tokens are generated for each instantiated family. If a family is not instantiated (for instance the SCADA family in Fig. 6), no token will be generated due to the fact that the values of the SCADA will be null. The number of tokens corresponds to the number of components that are initialized. It means that two tokens are generated in the place *client Family* for the model of the Fig. 6. For the DAS of the Fig. 7, 6 tokens are generated. Colours of each generated tokens are defined by the parameters values that are extracted using the function $InitClient()$. The same rational is used for the other components.

In other words, system integrators have to identify on the DAS architectures performances which have to be assessed and define their parameters. This step does not require to be a CPN expert. This result is captured with a dedicated tool using DSL. This description is exported to generate a formal description of the defined architecture using SGML such as XML file. This file is then parsed

Table 1. DAS parametrization

Fig. 6 DAS structure declaration
Val Client = ref [Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$; , cp3 = Val_{string}), Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$, cp3 = Val_{string})]; Fun InitClient ( ) = (!Client); Val PLC = ref [PLC (plcp1 = $Val_{integer}$)]; Fun InitPLC ( ) = (!PLC) Val SCADA = ref []; Fun InitScada ( ) = (!SCADA)

Fig. 7 DAS structure declaration
Val Client = ref [Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$; , cp3 = Val_{string}), Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$, cp3 = Val_{string})] Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$, cp3 = Val_{string})] Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$, cp3 = Val_{string})] Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$, cp3 = Val_{string})] Client (cp1 = $Val_{integer}$, cp2= $Val_{integer}$, cp3 = Val_{string})]; Fun InitClient ( ) = (!Client); Val PLC = ref [PLC (plcp1 = $Val_{integer}$), PLC (plcp1 = $Val_{integer}$), PLC (plcp1 = $Val_{integer}$)]; Fun InitPLC ( ) = (!PLC) Val SCADA = ref [SCADA(sp1 = $Val_{integer}$, sp2 = $Val_{integer}$)]; Fun InitScada ( ) = (!SCADA)

to provide the CPN parametrization and instantiation functions with the required values. Consequently, the generation of a formal CPN model from a DAS description can be considered as hidden for the system integrators and does not required any expertise about CPN modelling.

5.3 Performance assessment

The aim of this automatic model generation is to assess the performances of DAS architecture using CPN methodology. In order to evaluate its efficiency, an experimental reference has been deployed in laboratory (DAS architecture of the Fig. 6) in which performances have been measured. Experimentation focuses on the end-to-end delay, i.e. the time for the message to go from the client to a device through the PLC. This performance is monitored thanks to:

- a time stamp that is assigned to the tokens leaving the output buffer of a client; it requires the definition of an additional colour that records the time stamp value;
- a monitor 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 unique IDs defined for each instantiated component.

The architecture is split into four groups of devices; each group has a different scan rate. The operator workstation and engineering workstation send periodically requests to the PLC in order to get information retrieved by the PLC from the devices. These workstations also send non periodic requests to the devices through the PLC according to user's actions. All these actions are modelled using packet generator transition (deterministic laws produces a P-temporised behaviour while non periodic requests follow a Poisson process with a rate λ, i.e. duration between two successive requests is given by an exponential law with rate λ). The performance to be assessed are the end-to-end delay (emission of a request by the client and reception of an answer by the client, i.e. the sequence client-PLC-devices-PLC-client). The Table 2 shows the configuration of the architecture, values are in milliseconds. Note that the processing time of some components (PLC, network switch, ...) is randomly distributed on an interval according to a uniform distribution law; this stochastic feature is embedded in the CPN component library and is not parametrizable.

Table 2. Architecture configurations

	Type	Value
Device Group 1	scan time	25ms
Device Group 2	scan time	50ms
Device Group 3	scan time	75ms
Device Group 4	scan time	100ms
PLC	cycle time	50ms
Client 1 (operator workstation)	request period aperiodic request	100ms exponential law ($1/\lambda$ =100ms)
Client 2 (engineering workstation)	request period aperiodic request	150ms exponential law ($1/\lambda$ =100ms)

The Table 3 shows the performance results of the deployed architecture and the performances assessed by the simulation tool. All time are in millisecond, the performances are measured in terms or minimal, average and maximum time values. Confidence interval (CI) is given by half length for 95%. Time unit of the CPN model is the millisecond, the length of one simulation run is 20 minutes and 15 replications have been performed. The performances assessed by the simulation tool are close to the results retrieved from the laboratory. However, the maximum values are a little bit more important for the experimental run; it is mainly due to the firmware load on the PLC which is basically modelled as distribution law and to the limitation of our CPN model of the network.

6 Conclusion

This paper has demonstrated firstly that CPN is a relevant choice for modelling and assessing the performance of DAS architecture. Face to the pre-sales context constraints, the paper presents a relevant solution based on the automatic

Table 3. Performance assessment results

	Devices group 1				Devices group 2				Devices group 3				Devices group 4			
Laboratory results	Min	Avrg	Max	-	Min	Avrg	Max	-	Min	Avrg	Max	-	Min	Avrg	Max	-
Client1	78	129	180	-	103	116	205	-	128	179	230	-	153	204	255	-
Client2	81	154	228	-	106	167	253	-	131	205	278	-	156	230	303	-
Simulation results	Min	Avrg	Max	CI	Min	Avrg	Max	CI	Min	Avrg	Max	CI	Min	Avrg	Max	CI
Client1	76	127	178	1,9	104	115	203	2,2	124	178	229	2,7	151	203	256	3,3
Client2	79	153	226	2,5	107	169	251	2,9	131	204	280	3	152	228	307	3,8

generation of a CPN model. This automatic generation has 3 mains advantages: the CPN model holder and the library of CPN component models are done only once, a user without experience on CPN modelling can easily with parametrization generate CPN models through SGML description and finally, our approach allows testing a wide range of architectures without spending time on the modelling phase. Thereby this automatic model generation is a huge step forward for industrial companies to use CPN as model definition and simulation. However, paving the way towards a real industrial use of our approach requires further works:

- to face the large size of the parametrization and instantiation token for industrial application; indeed, this size may have a negative impact on the simulation time;
- to enrich the modelling of the network family which is currently limited to the modelling of switches with several ports.

References

1. Boussad, A., Saıd, A., Lesage, J.J.: Genetic algorithms for delays evaluation in networked automation systems. Engineering Applications of Artificial Intelligence 24(3), 485–490 (2010)
2. Brahimi, B., Rondeau, E., Aubrun, C.: Integrated approach based on high level petri nets for evaluating networked control systems. In: 16th Mediterranean Conference on Control and Automation. pp. 1118–1123. Ajaccio, France (2008)
3. Clarke, E., Grumberg, O., Somesh, J., Yuan, L., Helmut, V.: Progress on the state explosion problem in model. Informatics pp. 176 – 194 (2001)
4. Cruz, R.: A calculus for network delay, part i: Network elements in isolation. IEEE Trans. on Information Theory 37(1), 114–131 (1991)
5. Georges, J.P., Divoux, T., Rondeau, E.: Network calculus: application to switched real-time networking. In: 5th International ICST Conference on Performance Evaluation Methodologies and Tools. pp. 399–407. Paris, France (2011)
6. Goldfarb, C.F., Rubinsky, Y.: The SGML handbook. Oxford University Press (1990)
7. Jackman, S.: Estimation and inference via bayesian simulation: An introduction to markov chain monte carlo. American Journal of Political Science pp. 375–404 (2000)

8. Jasperneite, J., Neumann, P.: Switched ethernet for factory communication. In: 8th IEEE International Conference on Emerging Techonologies and Factory Automation, ETFA 2009. pp. 205–212. Antibes - Juan les Pins, France (2001)
9. Jensen, K.: An introduction to the theoretical aspects of coloured Petri nets. Springer (1993)
10. Jensen, K., Kristensen, L.M.: Coloured Petri nets: modelling and validation of concurrent systems. Springer Science & Business Media (2009)
11. Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with generalized stochastic Petri nets. John Wiley & Sons, Inc. (1994)
12. Meunier, P., Denis, B., Lesage, J.J.: Temporal performance evaluation of control architecture in automation systems. In: 6th EUROSIM Congress on Modelling and Simulation. Ljubljana, Slovenia (2007)
13. Milner, R., Tofte, M., Harper, R., MacQueen, D.: The definition of standard ml-revised the mit press. Cambridge, MA (1997)
14. Pinna, B., Babykina, G., Brinzei, N., Pétin, J.F.: Using coloured petri nets for integrated reliability and safety evaluations. In: 4th IFAC Workshop on Dependable Control of Discrete Systems, DCDS'13. pp. 19–24. York, UK (2013)
15. Seno, L., Vitturi, S., Zunino, C.: Real time ethernet networks evaluation using performance indicators. In: 14th IEEE International Conference on Emerging Techonologies and Factory Automation, ETFA 2009. pp. 1–8. Mallorca, Spain (2009)
16. Van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: An annotated bibliography. Sigplan Notices 35(6), 26–36 (2000)
17. Warmer, J., Vliet, H.: Processing sgml documents. Electronic publishing 4(1), 3–26 (1991)
18. Witsch, D., Vogel-Heuser, B., Faure, J.M., Marsal, G.: Performance analysis of industrial ethernet networks by means timed model-cheking. In: 12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2006. pp. 101–106. Saint-Etienne, France (2006)
19. Zwick, R., Walisten, T.: Combining stochastic uncertainty and linguistic inexactness: theory and experimental evaluation of four fuzzy probability models. International Journal of Man-Machine Studies 30(1), 69–111 (1989)