Temporal performance evaluation of control architecture in automation systems

Pascal Meunier, Bruno Denis, Jean-Jacques Lesage

TEMPORAL PERFORMANCE EVALUATION OF CONTROL ARCHITECTURE IN AUTOMATION SYSTEMS

Pascal Meunier, Bruno Denis, Jean-Jacques Lesage

LURPA, ENS Cachan, UniverSud Paris
61 av. du Président Wilson, F-94230 Cachan, France

pascal.meunier@lurpa.ens-cachan.fr (Pascal Meunier)

Abstract

The performances of automation systems are strongly linked to the performances of their control architecture. These architectures are the merger of a hardware structure – industrial computers and logic controllers connected to networks and fieldbuses – and of software components – implantation control functions. To manage these performances, the control engineer must evaluate them at each stage of the life cycle: from the project requirements to the setup stage, including the detailed design.
In this paper, we present a method which evaluates timed performances of networked automation system and which guides the engineer throughout the control architecture development. A modular design has been retained to model the dynamic behaviour of the control architecture using Timed Coloured Petri Nets. To begin we present the design of generic modular models and we explain how these modules are instantiated and assembled to build the dynamic model of a whole control architecture. Next we present some enrichments of the assembled model in order to simulate the behaviour of the control architecture. These complements are some event generators – to excite the models – and some event observers – to detect, to date and to log the relevant events. Then we present the simulation results of four different timed performances on the same architecture. These results take the form of histograms of 10,000 simulated delays for each evaluated performance. Finally, the method suggested for the model construction is validated. by confrontation between the results of model simulation and the measurement of the real control architecture.

Keywords: Control architecture, temporal performance, evaluation, simulation, timed coloured Petri nets.

Presenting Author's biography

Bruno DENIS obtained his Ph.D. in the field of control and automation system in 1994 at the University Nancy I. He currently holds the position of Associate Professor at the "Ecole Normale Supérieure de Cachan" close to Paris in France. His main research interest is the dependable control in automation system using approaches like simulation or formal method such as model-checking.

1 Introduction

A lot of performances of automated systems rely on temporal performances of their control architecture. Engineers must then be able to evaluate these performances during all phases of the system life cycle. The preliminary project is particularly crucial because in this phase a solution of control architecture is selected there with still not accurate information and sometimes the need for working quickly when the goal is to response to a call for tenders.

In order to meet this need, we developed a method of generic and modular modelling of control architecture using timed coloured Petri nets. This modelling takes into account data processing devices, communication devices and functions to be ensured in the system. Simulation of obtained models makes it possible to evaluate the expected temporal performances of modelled architecture.

In this paper we explain first, the context and objective of this work. After that, modelling principles will be presented. We will underline the generic and modular characteristics of our method. Simulation of the Petri net constructed models will be detailed in a second part of the paper. Finally simulation results will be discussed. A case study will be used to illustrate all the different aspects of the presented works and results.

2 Context and objectives

Control architecture includes a set of treatment devices (e.g. Programmable Logic Controllers, industrial computers). These devices compute necessary tasks (or functions) to control the plant.

These devices exchange information together by mean of communication channels (e.g. networks, field buses, crossed digital IO). A control architecture example is given Fig. 1.

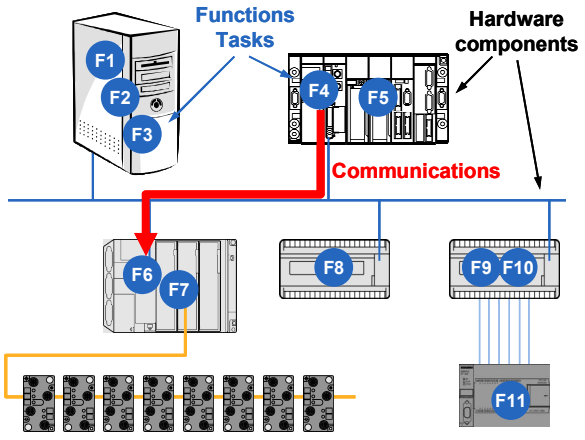

Fig. 1 Control Architecture Example

When control architecture is designed, some temporal performances are required for assessment, like reactivity (response time between an input occurrence and its causal output) or transmission delays… However, such temporal characteristics do not have deterministic value. To characterize them, it is thus necessary to identify a probability distribution function as well as statistic indicators (minimum, maximum, mean value...). For critical applications, the knowledge of the maximum value corresponding to the worst case delay is required. For all kinds of applications, the knowledge of delay distribution is useful. Fig. 2 presents an example of a delay of response time in control architecture.

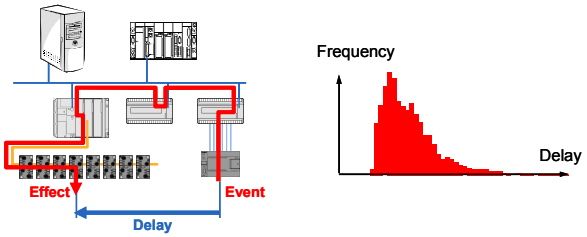

Fig. 2 Expected simulation results

3 Modelling

We developed a systematic assembly method of generic modular Petri net models. Evaluation of time performances is obtained by simulation of the assembled model. High level hierarchical timed and coloured Petri nets (K. Jensen [1]) have been retained for modelling. Hierarchy enables modular modelling; colours enable generic modelling while timed feature enables delays modelling. In order to achieve complex simulations CPNTools software [2] is used. In the following section we present construction of the generic models and then modelling of whole control architecture.

3.1 Structure of modular model

Modular modelling is well adapted to the development of the model of large systems, and the re-use of the modelling know-how in the next studies is facilitated. The most frequently used modules can be developed as generic models. The model design for new study case will be done both by instantiation of generic models gathered in a library and, if necessary, by the design of new specific models.

The main difficulty of modular modelling is the identification of relevant boundaries and relevant module interfaces. In practice the choice of the modules is based on the modularity of the modelled system. For the control architectures, the system behaviour is the result of interactions between hardware components (industrial computers, programmable logic controllers, fieldbuses...) and functions implemented into user's programs. We chose the modular aspect of hardware architecture to structure our model. Thus the base of the model will consist of a set of modules, where each module describes hardware component behaviour, and where modules are interconnected in a similar way to the

modelled components. The behaviour due to the software part of each hardware component of architecture (program of logic controllers) is modelled by distinct modules. A software module (functions implementation) will be only connected to the hardware module which processes it.

Fig. 3 illustrates with a simple architecture the principle which has been selected for the modularity. The three logic controllers PLC1, PLC2 and PLC3 and the fieldbus that constitute hardware architecture are modelled by four modules, and the program "Prog" implemented in PLC1 is modelled by its own module connected to the module of PLC1.

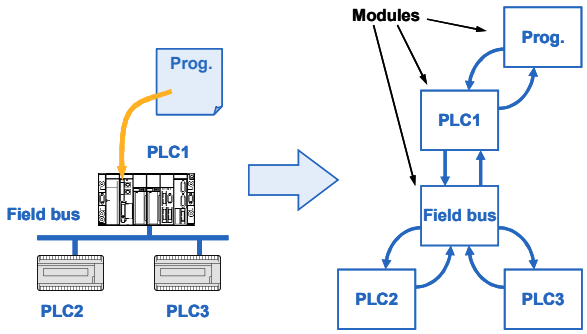

Fig. 3 The selected principle for modular modelling

To translate this modularity into a Petri net, the following rules were applied:

- Interfaces between the modules are made using places of Petri net
- Exchanged data in architecture are modelled by tokens
- In order to obtain generic models, and to avoid the increase in interface places with the increase in the number of senders and recipients of data exchanges, tokens that model data are coloured with their source and their destination.
- To interconnect the modules, rather than to add transitions and hub places between modules, the places interfaces having to be interconnected are amalgamated in a fusion set. It is a set of places so that anything that happens to each place in a set also happens to all the other places in the set. The places are then functionally identical. Such places are called fusion places in CPNTools
- To distinguish the interconnection part of the modules and the description part of the behaviour of the modules, two hierarchy levels are used. On the top level, modules appear as a single transition surrounded by its interface places. That gives a broad overview of the architecture model. By substituting transitions of this top level net with more detailed pages, we can bring detailed behaviour of modules (see Fig. 4).

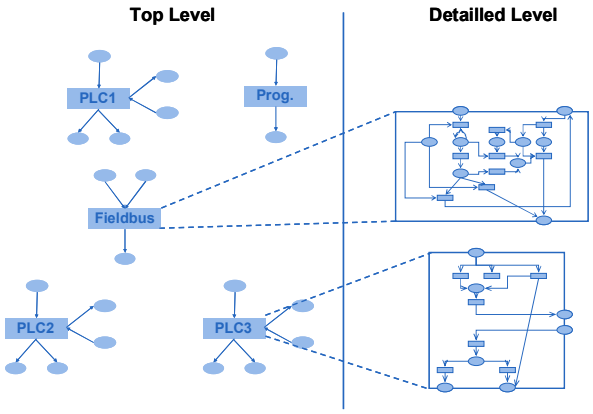

Fig. 4 Hierarchical structure

In the next section, we present the construction of generic model of modules.

3.2 Design of generic model of modules

In control architecture, the hardware components are often generic; it is then useful to design generic models of hardware components.

Each type of hardware components is associated to a dynamic behaviour. This behaviour is parameterized by a set of variables (cycle duration, delays…). To design a module, initially, it is necessary to define the module interfaces (inputs and outputs). Then in the second time, it is necessary to define the dynamic behaviour of the module.

To illustrate how a generic module is designed, an example is presented for a PLC. The chosen PLC is a widespread component of automation systems with cyclic and non pre-emptive behaviour. Its interfaces are connected with:

- its environment – the controlled system – via local input output signals
- one or more fieldbuses via network interfaces
- the functions (user's program).

To model hardware component, the determination of its dynamic behaviour is necessary. For that two approaches are possible: knowledge-based or identification-based. The documentation provided by the manufacturers gives information on the dynamic behaviour. But this behaviour is often simple and idealized. The experimental identification permits to construct a behavioural model very close to reality. But this behavioural model is often very complex, and difficult to obtain.

The presented example is built using the information described by Bhowal [4] for PLC free of pre-emptive tasks. As digital computer used for automation systems, a PLC scan its inputs, re-computes all the logic functions of the user program and updates its outputs. These three steps are repeated in a cyclic manner as presented Fig. 5.

Fig. 5 Cycle of a PLC

Fig. 6 shows the generic behaviour model designed. On the model boundary, five places provide the interface between the module of PLC and external behaviours. The place P1 located on the top collects all the PLC input data. The place P8 on the bottom-left contains the data transmitted by the PLC to its local output signals, while the place P9 on the bottom-right contains the data transmitted through a network. The two places P4 and P5 on the right border provide the interface with the modules describing the behaviour of the functions implemented into the PLC. On the right of the model, a sequence of three places P10, P11 and P12 and three transitions describes the cyclic feature of the PLC.

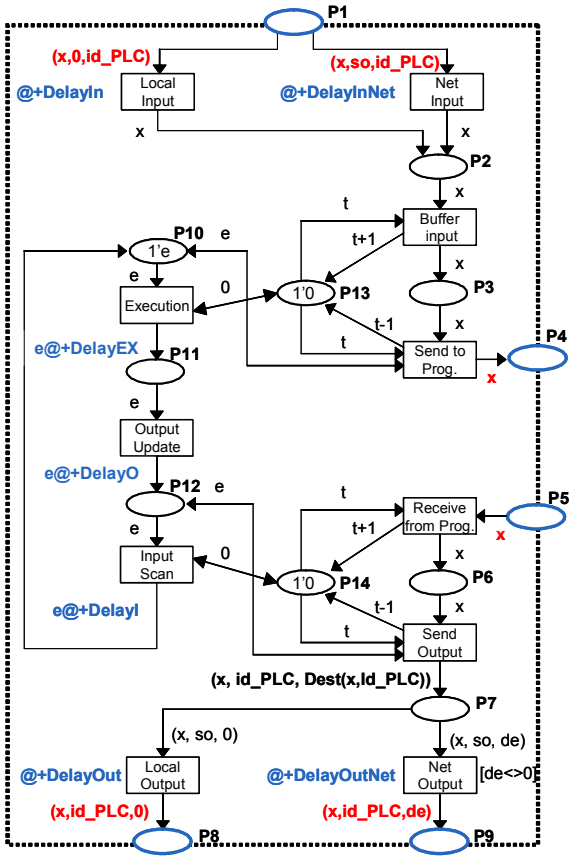

Fig. 6 Coloured Timed Petri Net of PLC behaviour

To achieve the correct behaviour, the Petri net model deals with two types of tokens for data modelling:

- tokens that model only data to be transmitted. For example (X) where "X" is the data
- tokens that model the data its destination and its source. For example (X, so, de) where "X" is the data, "so" is the identifier of source component and "de" is the identifier of destination component).

When a PLC input data produces a consequence on a PLC output, a data token flows the following way: P1, P2, P3, P4, P5, P6, P7 and finally P8 or P9. An input token in P1 goes to P2 then to P3 where it is buffered. All the data buffered into P3 are counted and the sum is stored into P13. As soon as the transition "Input scan" is fired the presence of a token in the place P10 ensures the transfer of all the data buffered into P3 to P4. Thus the counter into P13 falls down to zero and the transition "Execution" is fired. Without time-consuming, another module gets data in P4 to produce result data into P5. All the result data are buffered and counted into P6 and the sum is stored into P14. As soon as the transition "Output update" is fired the presence of a token in the place P12 ensures the transfer of all the data buffered into P6 to P7. Thus the counter into P14 falls down to zero and the transition "Input Scan" is fired. From P7, data are switched to the right output place P8 or P9 depending their destination. This switch is obtained thanks to the "Dest" function used in the weight expression of the arc between the transition "Send Output" and the place P7. This function gives the next destination of a data from its identifier and its current location (source).

In accordance with the syntax of CPNTools, time modelling can be associated as well with the transitions as with the outgoing arcs of the transitions. Delays to input or output data from or to other hardware component are modelled by timed transition while delays of each step of the PLC cycle are modelled by timed arcs.

At the time of the instantiation of this generic model the following parameters will have to be instantiated: the component identifier (Id_PLC), the input delays (DelayIn, DelayInNet), the output delays (DelayOut, DelayOutNet) and the PLC cycle delays (DelayI, DelayEX, DelayO).

Before adding this generic model of PLC behavior in a library, we have validated it by comparison between simulation results and measurements on real PLC. The validation of this generic model showed that it provided very realistic results but it costs a lot of computer time. This drawback is due to the cyclic firing of transition, even if no data is to be treated. We thus developed a "faster" model but less readable presented in [5]. In the next section, we will present the construction of a complete model.

3.3 Construction of control architecture model

Each of the behavioural models of the different components has been obtained according to the method described in section 3.1. The whole model of a specific architecture has now to be built by instantiation of generic model of components and then assembly of component models taking into account information flows.

To build a behavioural model of control architecture the following information is necessary: the description

of hardware architecture, the description of functional structure and the mapping of the functions into the hardware components. Fig. 7 presents a hardware architecture built around five PLC, one network of PLC and one network of remote input-output (AS-i). Fig. 8 presents the functional structure. For example, data O is computed by the functions F1, F3, F8, and F12 to produce the data O1, O2 and O3, while function F2 queries periodically the three functions F7, F11 and F16 for data exchange. Finally, Fig. 9 shows in which hardware component each function is located.

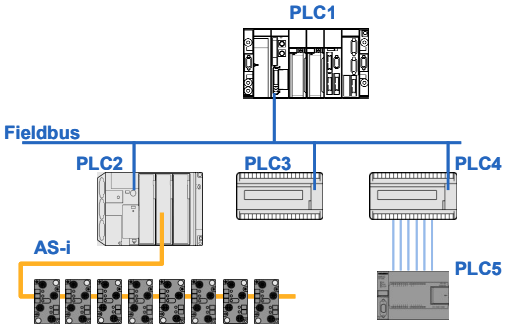

Fig. 7 Hardware architecture

The whole model of the control architecture has now to be built according the three following steps.

- Instantiation of all the generic modules useful for the modelling of the control architecture, and if necessary design of specific modules.
- Assembly of all modules by the creation of fusion sets with the interface places .
- Development of the function “Dest” from the functional structure and of the mapping of the functions into the hardware components.

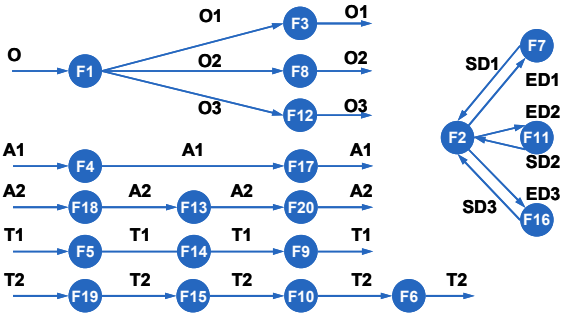

Fig. 8 Functional structure

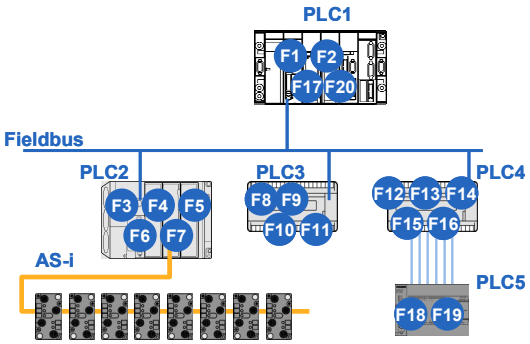

Fig. 9 Mapping of the functions into the hardware components

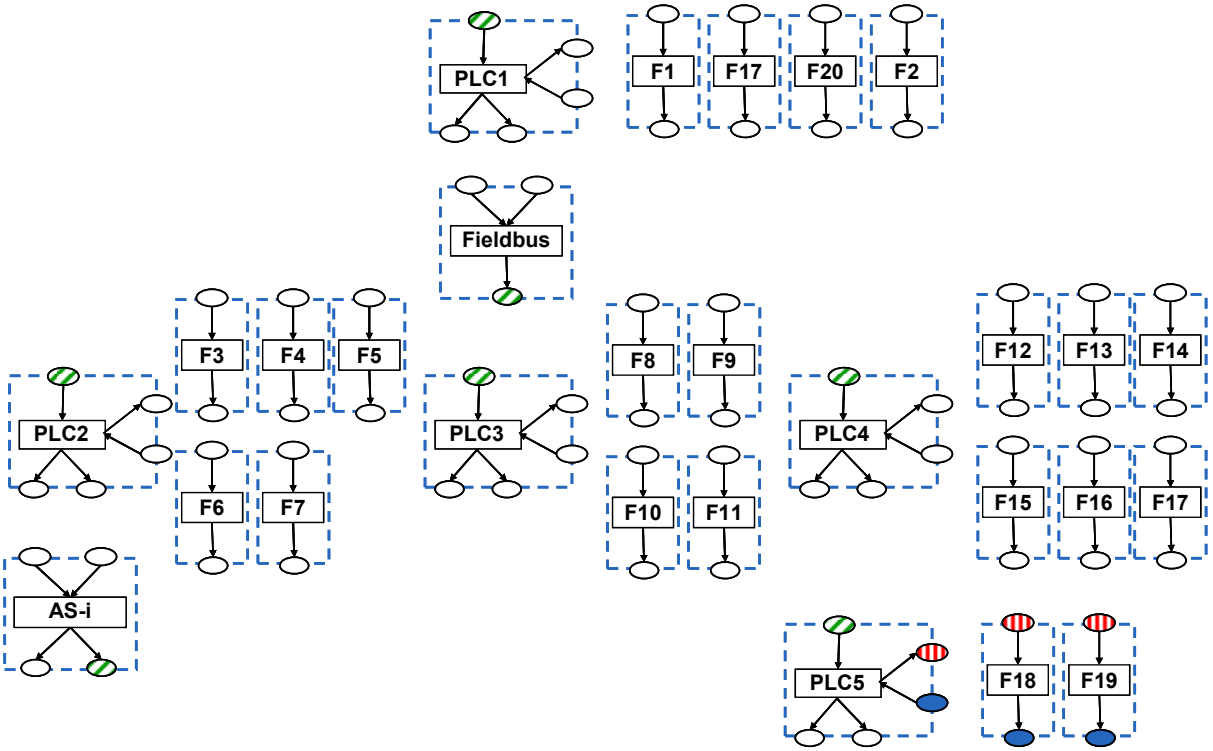

Fig. 10 Top-level of the hierarchy of the control architecture behaviour model

Fig. 10 presents the result of instantiation. Twenty seven modules have been instantiated from six generic modules: PLC, master-slave PLC fieldbus, remote input output fieldbus, regular data processing function, data polling function, data answering function. The instantiation task includes the setup of parameters such as delays and data transformation. Fig. 10. also present three fusions of the assembly task. The set of obliquely hatched places is the fusion place which models how the networks outputs are connected to the PLC inputs.

The set of three vertically hatched places is the fusion set which models the start of computation of functions F18 and F19 into PLC5 while the set of three filled places is the fusion set which models the end of computation of functions F18 and F19. Finally Tab. 1 gives a part of "Dest" function obtained by crossing the data of the Fig. 8 (functional structure) and Fig. 9 (mapping architecture).

Fig. 11 presents the whole Petri net model of the control architecture described Fig. 8, Fig. 9 and Fig. 10.

Tab. 1 "Dest" function definition table

Function inputs		Result
Data name	**Id of source component**	**Id of destination component**
O	Process	PLC1
O1	PLC2	PLC1
	PLC1	Process
SD1	PLC1	PLC2
ED1	PLC2	PLC1
…	…	…

4 Results

In this section an example is used for three purposes: first purpose aims to give experimental results of Petri net simulation for four temporal performances, second purpose aims to validate our choice of simulation of Timed Coloured Petri net, and third purpose aims to validate our modelling and simulation approach by comparison between simulation results and measurement data on a same architecture.

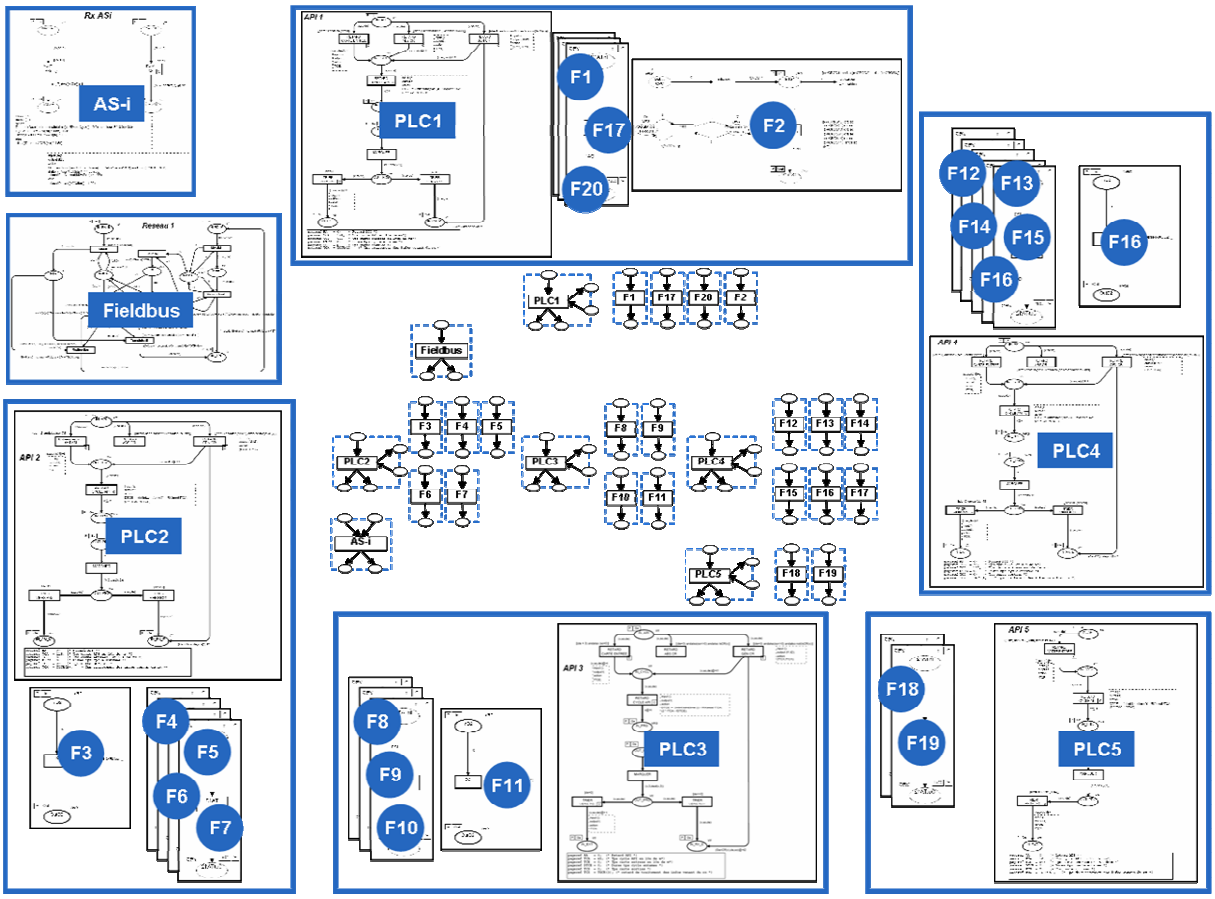

Fig. 11 Control architecture Petri net behaviour model

4.1 Simulation setup

Petri net simulation tools allow us to simulate marking evolution of the model from the initial marking. To excite the model built in section 3, we can either define a huge initial marking to model a scenario of events or add event generators to the model. We choose to define event generators. Fig. 12 presents a generator of three events (I1, I2, I3) according uniform distributions.

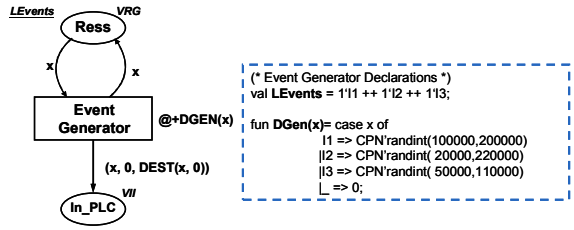

Fig. 12 Generic events generator

To collect performance results, rather than extract them from the full log file of fired transition we enrich the model with two non-intrusive event observers, one at the beginning of the evaluated delay (Fig. 13) and one at the end of the evaluated delay (Fig. 14). Each one store the token time stamp in a specific log file.

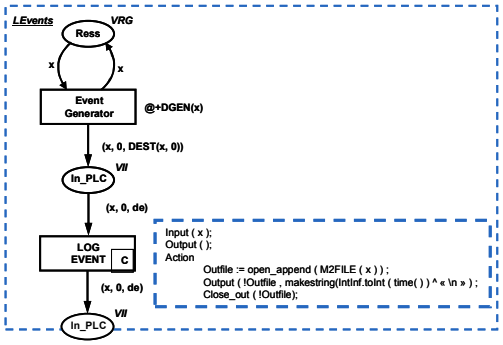

Fig. 13 Event observers

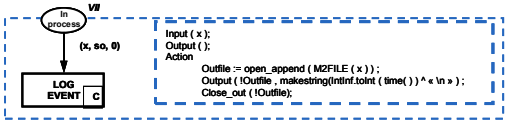

Fig. 14 Effect observers

4.2 Simulation results [6]

The simulation of the behaviour model of the control architecture presented Fig. 11 was carried out. The computing time to simulate one hour of operating architecture is 17 minutes. Four performances are studied:

- the response time of a bottom-up transmission of A1 data (alarm type),
- the response time of a top-down transmission of O1 data (order type),
- the response time of a horizontal transmission of T1 data, and
- delay between two reactions (O1 and O3) due to a top-down transmission.

To obtain at least 10,000 values for each performance, 12 hours of computation were necessary. Fig. 15 presents the simulation results.

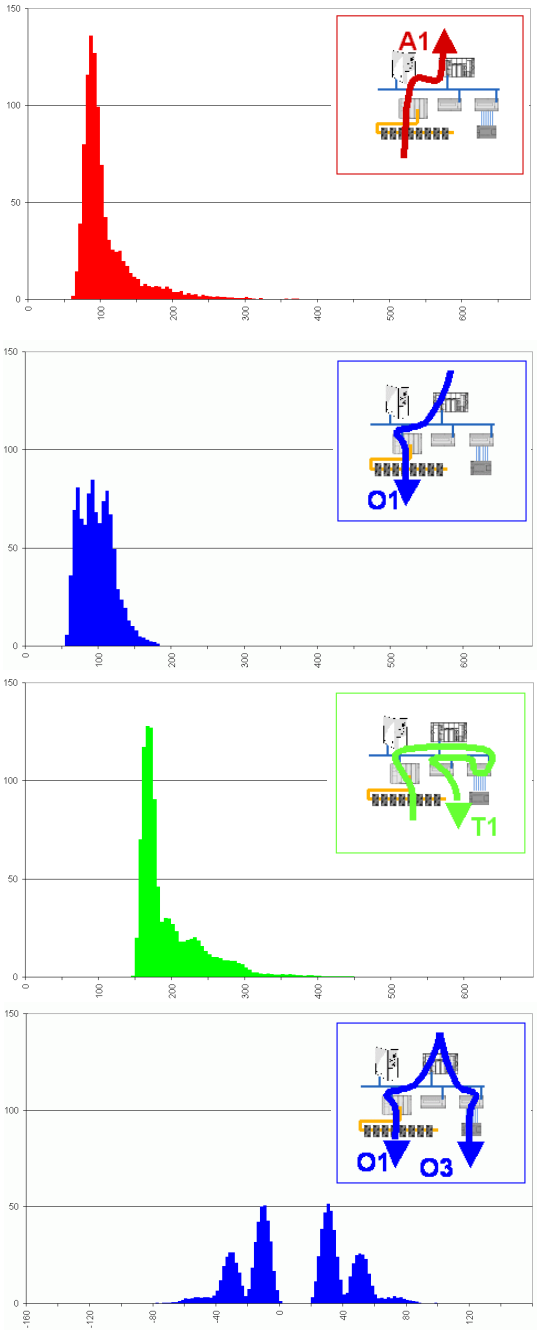

Fig. 15 Simulation results of four evaluated performances

4.3 First analysis: Validation of the choice of Coloured Timed Petri nets simulation

We chose to carry out a simulation of Coloured Timed Petri nets where all the model delays (except for model excitation) have deterministic duration. This choice has constrained us to design detailed models to be able to only deal with elementary delays of

constant duration. But was this the best choice to evaluate the temporal performances? Could we use Temporal Petri nets or Stochastic Petri nets to evaluate the same performances with less detailed models?

Let us take the simple case where E1 data cross two interconnected PLC as represented on Fig. 16 to produce the R1 reaction. If we consider an approach which models the treatment delay of a PLC by a uniform distribution, the evaluated performance will have the shape presented in Fig. 16.

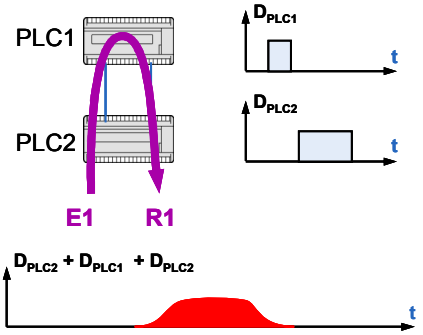

Fig. 16 Stochastic approach without correlation

That comes from the result of the sum of three random drawings. This approach does not take into account the correlation between these phenomena: the transmission crossing twice the same PLC, the second delay is not independent of the first. The second crossing is correlated with the first (Fig. 17).

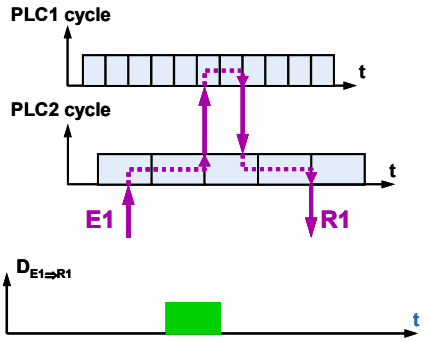

Fig. 17 determinist approach with correlation

The Stochastic Petri nets and the Temporal Petri nets cannot thus meet our requirements in evaluation for temporal performances. Our choice of Coloured Timed Petri nets for architecture modelling is validated.

4.4 Second analysis: Simulations results versus measurement results

To validate simulation results, it is necessary to check their accuracy and their relevance. For that, an experimental platform dedicated to the measure of the temporal performances has been used to measure on a real architecture. To check the accuracy of our simulation, we compare the statistics of the measurement results with simulation results.

Tab. 2 presents the absolute and relative deviations on minimum, average and maximum values of temporal performances A1, O1, O3-O1 and T1.

Tab. 2 Comparison of performance statistics obtained from measure and simulation

	A1	T1	O1	O1-O3
Minimum	+28ms (25%)	16ms (33%)	+4ms (7%)	22ms (24%)
Mean	-5ms (2%)	13ms (14%)	-1ms (1%)	-2ms (22%)
Maximum	239 ms (60%)	151ms (58%)	+17ms (9%)	-22ms (20%)

The comparison shows that simulation allows a very good estimation of the average and a rather good estimation of the minimal value. On the other hand the maximum value gives of worse results. That is because the maximum value of transmission is obtained when a conjunction of many phenomena appears. If during measurement or simulation, this conjunction is not observed, then the maximum value cannot be known.

Moreover, we compare simulation distributions and measure distributions. The covering rate between the distributions is used as comparison criteria. Fig. 18 presents coverings between the distributions of simulation and those of measurement.

The covering rates are very good. The small deviations are explained because PLCs are modelled with constant cycle times, whereas they are slightly variable in reality. Simulations give a very good estimation of the distributions of the temporal performances of control architecture.

5 Conclusion

In this paper, we have presented a modelling method of control architecture based on the assembly of generic modular models described by Timed Coloured Petri nets with deterministic delay. We showed then how to enrich the model to carry out an evaluation of temporal performances using simulation. Finally simulation results have been presented.

These results were confronted with measurements on real control architecture. This comparison have shown that our approach which take into account the correlation between modular behaviour gives results very close from the real. Only the estimation of the maximum value of performance is still difficult to correctly evaluate with non exhaustive evaluation method.

Our current works aim to use exhaustive methods of exploration like model-checking to obtain a guaranteed value of the maximum delay and to keep simulation approach to get performance distribution.

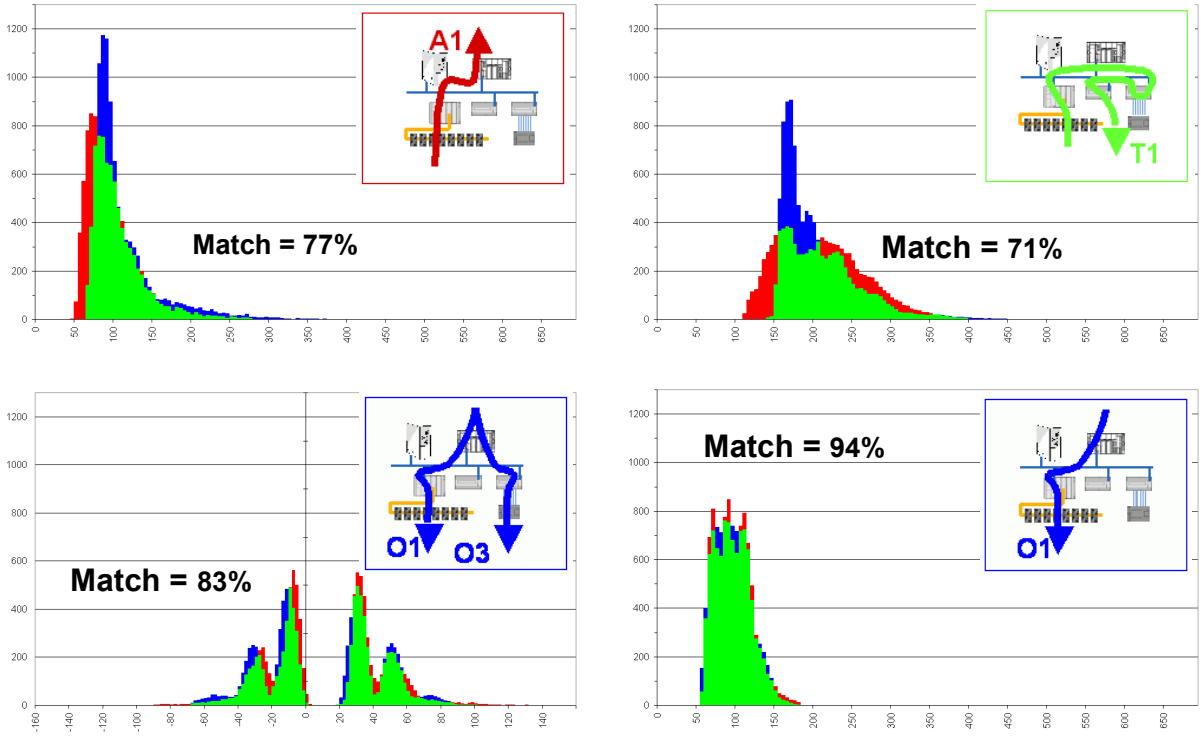

Fig. 18 Comparison of performance distributions obtained from measure and simulation – bleu union green bars are simulation results, red union green bars are measure results whereas green bars are intersections between simulation and measure results

6 Références

[1] K. Jensen. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. Monographs in Theoretical Computer Science Springer-Verlag 2nd corrected printing 1997. ISBN: 3-540-60943-1.

[2] Anne V. Ratzer, Lisa Wells, Henry Michael Lassen, Mads Laursen, Jacob Frank Qvortrup, Martin Stig Stissing, Michael Westergaard, Søren Christensen, Kurt Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. ICATPN 2003: 450-462.

[3] Søren Christensen, Jens Bæk Jørgensen, Lars Michael Kristensen. Design/CPN - A Computer Tool for Coloured Petri Nets. Proceedings in Tools and Algorithms for Construction and Analysis of Systems, Third International Workshop, TACAS'97, Enschede, The Netherlands, April 2-4, 1997, p 209-223.

[4] Prodip Bhowal, Rajib Mall and Anupam Basu. Estimating micro-PLC execution time for time critical system design, Journal of Systems Architecture, Volume 45, Issue 14, July 1999, Pages 1245-1248.

[5] Pascal Meunier, Bruno Denis and Jean-Jacques Lesage. Comparison of different modelling approaches in simulation of Programmable Logic Controller. IFAC conference on Control Systems Design, CSD'00, june 2000, Bratislava (Slovak Republic).

[6] Pascal Meunier. Evaluation de performance d'architectures de commande de systèmes automatisés industriels, PhD thesis of "Ecole Normale Supérieure de Cachan" in electrical and automation engineering, 209 p., mars 2006