# SAFETY ANALYSIS DURING THE CONTROL ARCHITECTURE DESIGN OF AUTOMATED SYSTEMS

Pascal Meunier, Bruno Denis, Jean-Jacques Lesage

# SAFETY ANALYSIS DURING THE CONTROL ARCHITECTURE DESIGN OF AUTOMATED SYSTEMS

**P. Meunier, B. Denis et J.J. Lesage**

*Laboratoire Universitaire de Recherche en Production Automatisée, ENS de Cachan,*
*61 avenue du Président Wilson, F-94235 Cachan Cedex, France*
*{meunier, denis, lesage}@lurpa.ens-cachan.fr - http://www.lurpa.ens-cachan.fr*

**Abstract**: To design safety automated systems, the designer must study safety according to various points of view. This paper deals with safety analysis at the stage of control architecture design. The benefits of our work is the taking into account of both software and hardware aspects of the control to analyze the system safety. We illustrate our matter with an example: a cable car system. Several control architectures are considered. Firstly, the reliability of a critical function is analyzed. This analysis will support the choice of an architecture rather than an other one. Secondly, the performances of suggested architectures are evaluated. This evaluation obtained with a Petri nets simulation will allow us to qualify the critical functions.

**Keywords**: command and control systems, safety analysis, performance evaluation, architectures, Petri-nets

**Relevant themes**: Design for reliability and safety (Evaluation of safety systems)

## 1. Introduction

To design safety automated systems, the designer must study safety according to various points of view. Many papers in the mechanical engineering deal with the safety of operating part of automated systems. The control system is an other relevant point of view of the safety according industrial control engineers [Boudennec C. and al. 1998]. Works on control system safety focus on functional structures of control [Kiencke U. and al. 1999], on components (such as logical controller, fieldbus,...) [List V. and al. 1998] or sometime on redundancy component associations [Gulner J. and I. Theis 1999]. Our work deals with the control architecture at the design stage. Control architecture is the result of the projection of a functional structure onto a component structure. The benefits of our work is the taking into account of both software and hardware aspects of the control system to analyze the system safety. In the session 2 an example of automated systems with safety requirements is presented. It is a cable car system. Several possible control architectures are considered. Session 3 is devoted to the analysis of the availability of a critical function. This analysis will lead us to sort control architectures relating the reliability criteria. Lastly, in session 4, the performances of control architectures are evaluated. This evaluation obtained with a Petri nets simulation will allow us to qualify the critical functions.

## 2. Presentation of the example

### 2.1. The operative system

The support of our study is a cable car (fig. 1). It allows the transport of passengers between two stations. The passengers take seat in cabins that are sus-

pended with a carrying cable. A second cable ensures the traction of the cabins. The two cables form a closed loop between the two stations. Wheels placed on pylons ensure the guidance and the drive of the cables. The power transmission system of the driving wheel is double. In normal operating mode, motion is obtain by an electric motor. In recovery operating mode, motion is obtain by a thermal engine.

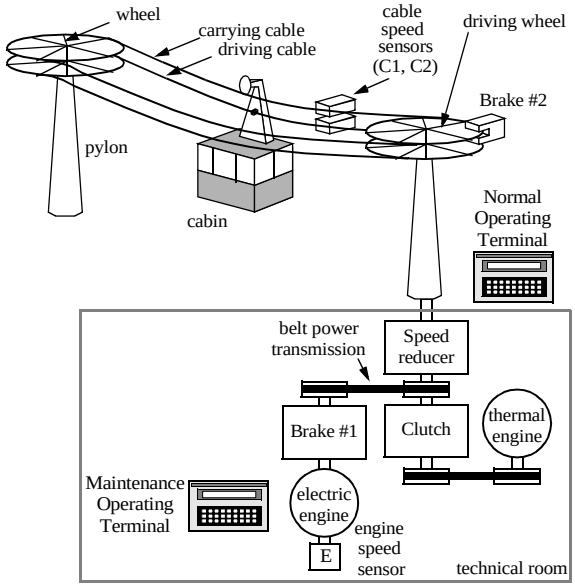



Fig. 1. general view of the system

Brakes #1 and #2 are actuated if a problem is detected on speed of cables. Three speeds are monitored: driving cable speed with C1 sensor, carrying cable speed with C2 sensor and electric engine speed E with sensor.

Two operating terminals manage the human-machine interface. The first one is in the control room and second is placed in the technical room.

2.2. Required control functions

The control system of the cable car must ensure the five following tasks:

- Normal Operating Mode (NOM)
- Recovery Operating Mode (ROM)
- Safety Task (ST)
- Normal operating Terminal Management (NTM)
- Maintenance operating Terminal Management (MTM)

2.3. various possibilities of control architecture

Three control architectures are studied. The first one uses two Programmable Logical Controllers (PLC). The first PLC manages NTM and NOM tasks and the second one manages all the remaining tasks (fig. 2).

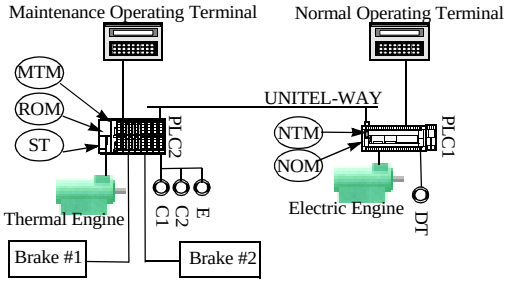



Fig. 2. architecture with to PLC (2P)

The second control architecture uses three PLC. The two first have the same functions as for the architecture 2P. The third PLC manages in redundancy manner the safety task ST. Each PLC managing the safety task ST is connected to its three owned speed sensors i.e. speed data are collected by redundant speed sensors. At last, a safety relay send braking order with signals coming from these two PLC (fig. 3).

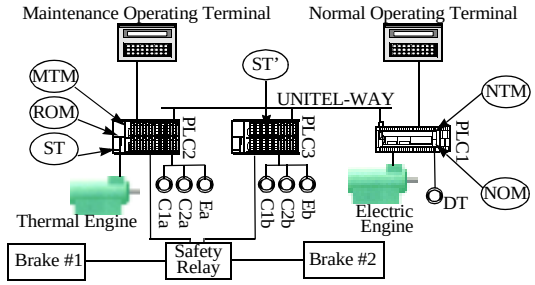



Fig. 3. architecture with three PLC and six sensors (3P6S)

The third control architecture looks like the second one. But there are not redundant sensors. A fieldbus allows PLC to share data coming from sensors (fig. 4).

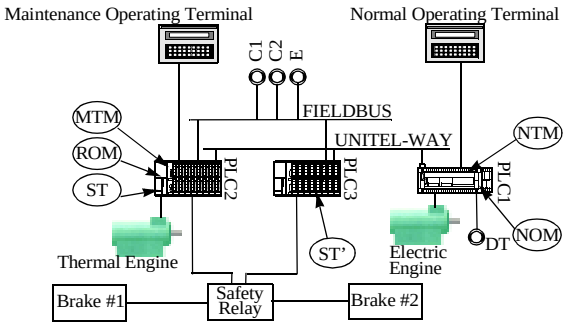



Fig. 4. architecture with three PLC and three sensors (3P3S)

3. Choice of control architecture vs. system reliability

In this part, we will show implications of an architecture choice on reliability. First of all, we will present a method to build reliability diagrams. Then we will present a comparison between various architectures.

3.1. Construction of reliability diagrams

We will study the reliability of the capacity of the «safety task» (ST) to send an order the brake #1. The method will be detailed for the "2P" architecture including two PLC. Only reliability diagrams will be presented for the other architectures. In order to build the reliability diagram, it is necessary to know which components are involved in the function implementation (fig. 5).

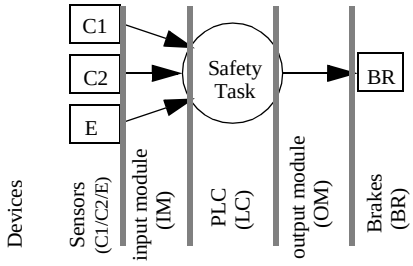



Fig. 5. Functional point of view and hardware implementation of "Safety task"

Now, a description of the task behavior is needed i.e. which inputs are required to produce outputs. A simple Petri net model is use to describe the input-output transformation. The model of the safety task is presented on fig. 6. The three inputs must be set together to produce the output.

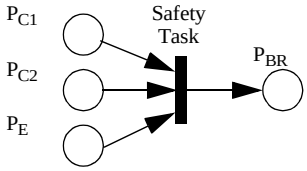
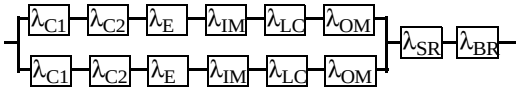



Fig. 6. Petri net model of the safety function

Then, the sensors reliability are draw in series on the reliability diagram. The other hardware devices (input module, PLC, output module and brake) appear successively. Their reliabilities will be also organized in series pattern. The fig. 7 describes the reliability diagram obtained.

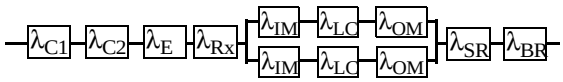

Fig. 7. reliability diagram for the control architecture 2P

Fig. 8 and fig. 9 present reliability diagram obtained with the two other control architectures.




with $\lambda_{C1a} = \lambda_{C1b} = \lambda_{C1}$ $\lambda_{IM\ PLC1} = \lambda_{IM\ PLC2} = \lambda_{IM}$
$\lambda_{C2a} = \lambda_{C2b} = \lambda_{C2}$ $\lambda_{LC\ PLC1} = \lambda_{LC\ PLC2} = \lambda_{LC}$
$\lambda_{Ea} = \lambda_{Eb} = \lambda_E$ $\lambda_{OM\ PLC1} = \lambda_{OM\ PLC2} = \lambda_{OM}$

Fig. 8. reliability diagram for the control architecture 3P6C




with $\lambda_{IM\ PLC1} = \lambda_{IM\ PLC2} = \lambda_{IM}$ $\lambda_{OM\ PLC1} = \lambda_{OM\ PLC2} = \lambda_{OM}$
$\lambda_{LC\ PLC1} = \lambda_{LC\ PLC2} = \lambda_{LC}$

Fig. 9. reliability diagram for the control architecture 3P3C

3.2. Comparison between architectures

Two criteria are used to compare control architectures. These criteria are the MTTF of functions and the cost of architectures. The MTTF are calculated with the reliability of the various parts of control architectures. These reliabilities are obtain with a reliability diagrams and rules of composition according to serial or parallel configurations in diagram (Pagès and Gondran, 1980). For the control architecture using two PLC, we get:

$$R_{ST}(t) = \prod_{components} R_i(t) \text{ with } R_i(t) = e^{-\lambda_i t}$$

to model the reliability of electronic devices.

Then:

$$R_{ST}(t) = e^{-\lambda_{SF} t} \quad \text{with} \quad \lambda_{ST} = \sum_{components} \lambda_i$$

$$MTTF_{2P} = \int_0^\infty R_{ST}(t) = \frac{1}{\lambda_{ST}}$$

The MTTF of other architectures ($MTTF_{3P6S}$ and $MTTF_{3P3S}$) are calculated in a similar way. The more MTTF is important the more architecture is reliable. However, the MTTF depends on the choice of the components of architecture.

Let us observe the MTTF of the control architecture with 3 PLC and 6 sensors (3P6S):

$$MTTF_{3P6S} = \frac{2}{\lambda_{SR} + \lambda_{BR} + \lambda'} - \frac{1}{\lambda_{SR} + \lambda_{BR} + 2\lambda'}, \text{ with}$$

$$\lambda' = \lambda_{C1} + \lambda_{C2} + \lambda_E + \lambda_{IM} + \lambda_{LC} + \lambda_{OM}$$

It is noticed that the difference with the $MTTF_{2P}$ is in the failure rate of safety relay ($\lambda_{SR}$). According to the value of $\lambda_{SR}$, $MTTF_{2P}$ could be higher (or lower) than $MTTF_{3P6S}$. We define the ratio $\Gamma$ in the following way:

$$\Gamma(\lambda_{SR}) = \frac{MTTF_{3P6S}}{MTTF_{2P}}$$

$$\Gamma(\lambda_{SR}) = \frac{(\lambda' + \lambda_{BR})(3\lambda' + \lambda_{SR} + \lambda_{BR})}{(\lambda_{SR} + \lambda_{BR} + \lambda')(\lambda_{SR} + \lambda_{BR} + 2\lambda')}$$

When $\Gamma$ is equal to 1, the $MTTF_{2P}$ is equal to $MTTF_{3P6S}$. The value of $\lambda_{SR}$ for $\Gamma$=1 is define by :

$$\lambda_{SR}^2 + \lambda_{SR}(\lambda_{BR} + 2\lambda') - \lambda'(\lambda_{BR} + \lambda') = 0$$

There is two solutions ($\lambda_{SRA}$ and $\lambda_{SRB}$) for this equation. But only $\lambda_{SRB}$ is possible ($\lambda_{SRA}$ is lower than 0).

$$\lambda_{SRB} = \frac{\sqrt{(\lambda_{BR}^2 + 8\lambda'\lambda_{BR} + 8\lambda'^2)} - (\lambda_{BR} + 2\lambda')}{2}$$

If the $\lambda_{SR}$ value is near zero, the $\Gamma$ value tends towards :

$$\Gamma_{\lambda_{SR} \to 0} = \frac{(3\lambda' + \lambda_{BR})}{(2\lambda' + \lambda_{BR})} = 1 + \frac{\lambda'}{(2\lambda' + \lambda_{BR})} > 1$$

So, if the value of $\lambda_{SR}$ is lower than the value of $\lambda_{SRB}$ then $MTTF_{2P}$ is lower to $MTTF_{3P6S}$. But the choice of the $\lambda_{SR}$ value is linked with the cost of the Safety Relay (SR). A lower $\lambda_{SR}$ value implies a higher Safety Relay cost. So the choice the control architecture is thus a compromise between the reliability of the architecture and its cost.

## 4. Study of the system safety by of performance evaluation of control architecture

In this session, we will present the various stage of the evaluation. Initially, we will show how to model the dynamic behavior of control architecture. Then, we will enrich obtained model in order to observe the performances during simulation. Lastly, we will expose the results of simulation and theirs analyses.

### 4.1. Dynamic behavior modeling of architectures

We will proceed in two stages. We will start with the modeling of the dynamic behavior of the software elements and the hardware elements of control architecture. Then we will assemble the various models previously selected to constitute the model of the control architecture.

The dynamic model of a software element must describe how the relevant flows of data are sent. It is considered that the flows, which take part in the load of the control architecture, are relevant. Any flood with an external element is not relevant except if it is explicitly mentioned in a performance criterion. Let us take the case of the task "Normal Operating Mode" (NOM) (fig. 10).

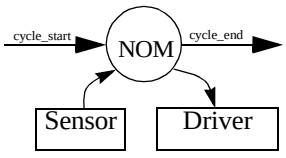



Fig. 10.task "Normal Operating Mode" (NOM)

Only flows "cycle_start" and "cycle_end" are relevant. Fig. 11 presents a simulation model of the dynamic behavior of the NOM Task. Design/PN is the used simulator tool [Jensen K., 1997]. Delay DCC corresponds to the duration NOM task, i.e. the time to move lower cabins to the upper station. Only these flows are represented in the PN model (fig. 11).

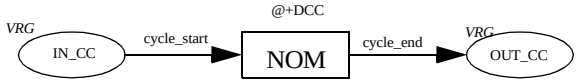



Fig. 11.Colored and Timed PN model of NOM task in Design/PN tool

This PN model behavior is: when a *cycle_start* token appear in the *IN_CC* place at the *d* time, the *NOM* transition occur. It remove the *cycle_start* token from the *IN_CC* place and produce a *cycle_end* token in the *OUT_CC* place. This token is not available until the *d+DCC* time. When the *cycle_end* token will be available, the Normal Operating Mode cycle will finish.

For the hardware component, a library of dynamic behavior model has been defined. Each model corresponds to the behavior of a class of hardware component. Two component of a same family accept the same model but their characteristics will be different (cycle time...).


Fig. 12.Colored and Timed PN model of a PLC with a periodic cycle time in Design/PN tool

The models used have been designed to be simulated quickly (few places and few transitions) while respecting the behavior expected of hardware component. The fig. 12 presents the model of a PLC with a periodic time cycle.

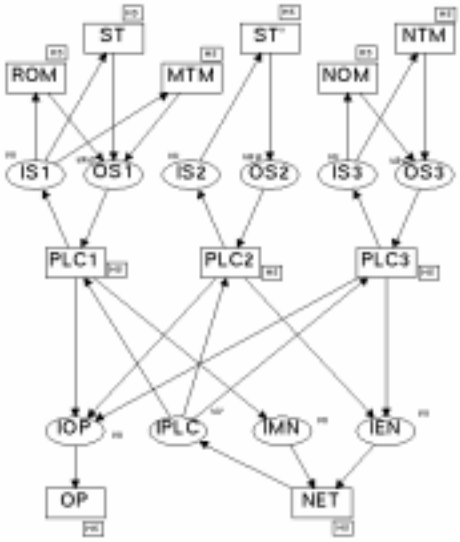



Fig. 13.hierarchical PN model of the control architecture using 3 PLC and 6 sensors

The PLC's PN model behavior is describe as follow: When a Input token *x* (x is any element of the VRG set) appear in the *IN_PLC* place at the *d* time, the *PLC* transition occur. It remove the *x* token from the *IN_PLC* place and produce a *x* token in the *OUT_PLC* place. This token is not available until the *d+delay* time. When the *PLC* transition occur, the *delay* value is calculated with the *PLC* transition code (dotted frame in fig. 12). When the token will be available the PLC will send an x information to his output module.

When all the needed models of various elements of control architecture are built or selected in a library, the behavior models of architecture is obtain by assembling. This assembly is made in order to respect associations between the software elements and the hardware components of architecture. The fig. 13 represents hierarchical PN model of the control architecture using 3 PLC.

4.2. Enhancement of the simulation model

The model previously created is ready to be simulated but it does not allow yet measuring performances of control architecture. We add code lines in transitions (possibility allowed by the Design/PN simulator). Let us consider the measurement of a response time of a message into control architecture. In order to do that, we put «probe codes» at the beginning transition (where an event appear) and at ending transition (where its effect occurred). For example, the performance studied is the transmission delay between a speed cable problem and the brakes activation (fig. 14). The probed message is the speed cable problem which appear in place $P_1$ in the Petri net. The brakes orders appear in place $P_{j+1}$. When the $T_{CBp}$ transition occurs, cable speed sensors had detected a problem. When the $T_{BR}$ occurs, activations orders had been sent to the brakes. To qualify the control architecture, the transmission delay must be low to avoid the cable break. For the example, it must be lower than 0.1 s.

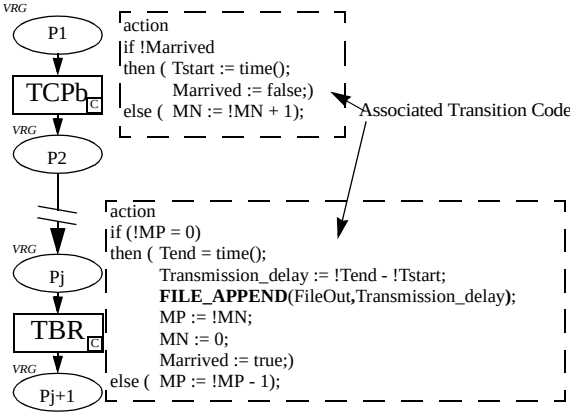



Fig. 14.Transitions code for the measure of transmission delay between a cable problem detection and a braking order.

The fig. 15 represents the behavior of the transitions code execution. On the first two chronograms, an arrow represents a message which occur a transition ($T_{CPb}$ or $T_{BR}$). A bold arrow represent that the message is probed. The transmission delay is the delay between a *Tstart* time and a *Tend* time.

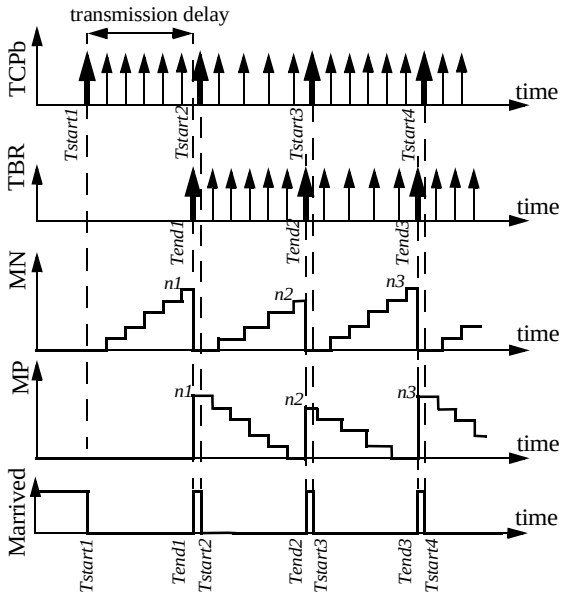



Fig. 15.Transitions code behavior

The probed message n°i (at the time $Tstart_i$) is the message which follows the $Tend_{i-1}$ time. To know which message is probed at the $T_{BR}$ transition, we count the no-probed messages at the $T_{CPb}$ transition. The number of no-probed messages at $T_{CPb}$ transition between probed message n°i and n°i+1 is counted by the *MN* variable. And for the $T_{BR}$ transition, the *MP* variable is used. The *Marrived* boolean variable is used to indicate when the *Tstart* time has to be memorized. The *Tend* time is memorized when a message occur at the $T_{BR}$ transition and the *MP* variable is null. The function *FILE_APPEND(output,x)* write the value *x* at the bottom of the file *output*. This file is used to analyze the simulation.

These transition codes log relevant firing transition, but do not change the firing rules of the Petri net model. Thus, the model of dynamic behavior of control architecture is not modified.

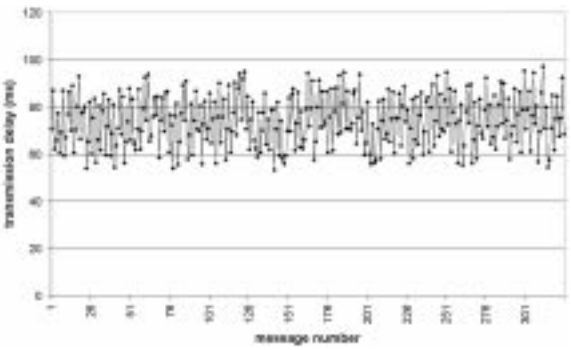


Fig. 16.Simulation results for the 2P control architecture

4.3. Simulation and results

When a response time is studied, the simulation of control architecture gives a list of time of response (fig. 16). To analyze and compare these results, they are formatted as a histogram and basic statistical data are calculated (minimal, maximum, and average values) (fig. 17).

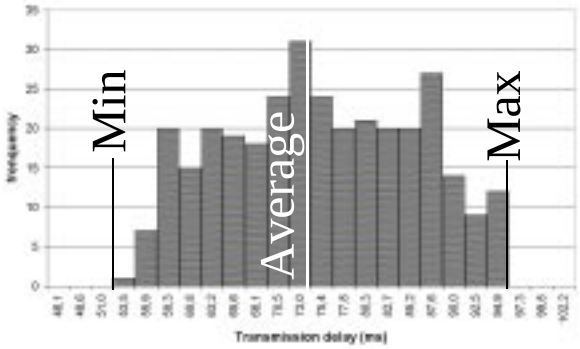



Fig. 17.Formatted results (histogram) for the example

These formatted data allow an easy comparison between the performances of various studied control architectures. For the example, the maximum value of the transmission delay is equal to 97 ms, so the evaluated architecture reacts in a compatible time with the safety level required in case of cable problem. If there is more than one compatible architecture, the results of simulation allow the selection of the control architecture with other criteria. For example, the criteria could be the lower average transmission delay for the lower control architecture cost.

## 5. Conclusions

This paper has presented two kinds of safety analysis of control architecture in order to assist the designer to select the best compromise. Analysis uses classic techniques such as reliability diagram or performance analysis by simulation of a Petri net model of a dynamic behavior. But in both cases, functional structure and component structure of control architecture are taken into account simultaneously. Then, our work allows designers to select the best control architecture during the early stage of automated system design. The works in progress are the comparison of the simulation results for various control architecture and the conception of hardware components behavior models.

## 6. References

Bodennec C., C. Jourdain, C. Mazuet, R. Garnier and Perez (1998). Dependability of safety critical systems: complementary of probabilistic and formal method. In: *Proceeding of INCOM'98*, Nancy, France.

German R. and A. Heindl (1999). Performance evaluation of IEEE 802.11 wireless LANs with stochastic Petri net. In: *Proceeding of PNPM'99*, Zaragoza, Spain.

Guldner J. and I. Theis (1999). Comparison of redundancy for safety relevant automotive control system. In: *Proceeding of ECC'99*, Karlsruhe, Germany.

Jensen K. (1997) Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. Monographs in *Theoretical Computer Science*, Springer-Verlag.

Kiencke U., R. Majjad and S. Kramer (1999). Modeling and performance analysis of a hybrid driver model. In: *Control Engineering Practice*, *7* (8), pp. 985-991

List V., A. Chovin, P. Coat, G. Renard and P. Soeters (1998). Safety integration in distributed automation systems, fieldbus - dependability. In: *Proceeding of INCOM'98*, Nancy, France.

Pagès A. and M. Gondran (1980). *Fiabilité des systèmes*, pp. 71--73. Collection de la Direction des Etudes et Recherches d'Electricité de France. EYROLLES, 1980.