# METHODS FOR SAFE CONTROL SYSTEMS DESIGN AND IMPLEMENTATION

Jean-Marc Faure, Jean-Jacques Lesage

# METHODS FOR SAFE CONTROL SYSTEMS DESIGN AND IMPLEMENTATION

**Jean-Marc Faure [1,2], Jean-Jacques Lesage[2]**

*[1] ISMCM-CESTI, 3 rue Fernand Hainaut*
*F 93407 Saint-Ouen Cedex*
*[2] Laboratoire Universitaire de Recherche en Production Automatisée*
*ENS de Cachan, 61 avenue du Président Wilson, F 94235 Cachan Cedex*
*{faure, lesage}@lurpa.ens-cachan.fr*

Abstract: This paper is the introductory one of the *Safe control systems* session. A classification of methods contributing to control systems safety is proposed in order to place the five other papers of this session and to show that they are complementary. This classification is based on a life-cycle criterion. Focusing then on discrete event systems safety, we point out the relationships between state space synthesis and analysis and system safety. This enables a more formal approach of safe control design and implementation.

Keywords: Off-line safety, On-line safety, Life-cycle, Discrete event systems, Discrete state space.

## 1. INTRODUCTION

Safety is a major industrial issue. Since the beginning of the last century and especially since the 50s, numerous works aiming at assessing and at improving the reliability and the availability of mechanical or electrical components or systems have been performed. These works dealt with process safety and have produced results like FMECA (Failures Modes, Effects and Criticality Analysis), HAZOP (HAZard and OPerability) analysis, Fault Tree Method,…

The growing demand for automated systems has led to also take into account control systems safety. Control system safety is twofold. First a control system has to be safe, by not generating hazardous events. Moreover a control system performs often a safety function, by monitoring a physical process, detecting failures and potentially hazardous deviations in order to undertake reactive actions aiming at putting the system in a safe state or to warn an operator. With regards to this safety service the control system shall be dependable.

Some industrial fields, like transport and critical industries (chemical processes, thermal and nuclear power plants, oil industry, …) seem to be more particularly concerned by safety. But in fact all the fields (manufacturing processes, building, bank, …) have nowadays to carefully consider the safety of their control systems.

It is the reason why several specific standards dedicated to the different application fields have been developed, e.g. standards for train monitoring and railways signalling systems, for power conversion, for energy transport and distribution. On the opposite side, the recent IEC 61508 standard (IEC 61508, 1998-2000) proposes a generic model which can be applied to all Electrical /Electronic/ Programmable Electronic (E/E/PE) safety-related systems. This standard includes the definitions of two kinds of safety: functional safety and safety of an equipment, as well as a safety life-cycle model which provides a framework for addressing all relevant phases associated with safety-related systems. The parts 2 and 3 of this standard are the most interesting from a control engineering viewpoint. They deal respectively with the architecture of the E/E/PE system and with the life-cycle of the software. Another key feature of this standard is the concept of Security Integrity Level (SIL) that enables to assess the reliability of a safety function. The SIL of a safety function characterises the probability of failure to perform its design function on demand and the maximum rate of failure in a continuous mode of operation.

Obviously this standard has to be taken into account when designing an industrial control system. Nevertheless, and even if its parts 6 and 7 are dedicated respectively to application guidelines and to an overview of techniques and measures, it does not intend to provide precise methods and tools enabling to design and to implement any control system, e.g. to define and to assess a control architecture, to elaborate a safe control model, to verify a control software module.

During the last years, several research works have been performed happily so as to develop methods contributing to control systems safety. A lot of them have produced very interesting results. It is the reason why, considering on one hand the strong industrial demand for safe control systems and on the other one the recent results of research works, it has been decided to join together some of these works within an invited session entitled *Safe control systems*. This session does not claim to be exhaustive, but merely aims at federating various complementary research works that all contribute to safe control systems design and implementation. In order to limit the scope of this session, we will focus mainly on Discrete Event Systems (DES) control though some of the presented works can be applied to continuous or hybrid systems.

## 2. SAFETY AND CONTROL SYSTEM LIFE-CYCLE

Considering the great amount of works developed in order to design and implement safe control systems, classification criteria shall be defined to organise these works. Laprie (Laprie *et al*,1996) for instance proposes a classification based on the expected objective. The safety-related works are arranged in four categories:

- Fault prevention methods
- Fault forecasting methods
- Fault tolerance methods
- Fault removal methods.

Fault prevention methods are not strictly specific to safety. Indeed they aim at organising processes and are rather related to system engineering. Functional analysis, project organisation and management methods, business process engineering are examples of such methods. Training of human operators is also a good way to prevent fault.

Fault forecasting methods can be split in two categories: ordinal assessment methods, like FMECA, and probabilistic assessment methods, that provide numerical values of safety attributes (reliability, availability). These last ones use models such as fault trees, Markov chains, stochastic Petri nets.

Fault tolerant methods are based on the assumption that the occurrence of a failure should not stop the system operation. When a failure occurs, three steps are then sequentially performed: failure detection, fault diagnosis and recovery. These methods fit well with systems owning some flexibility degrees and/or redundancies.

Fault removal when designing and implementing a system consists in verifying the results of the different activities (models, drawings, schemata, programs, sub-systems), by tests, simulation techniques or formal methods. During the operation of the system, fault removal is performed by corrective and preventive maintenance.

In spite of the relevance of the previous classification, it has not been adopted for this session. In order to point out that safety should be a permanent concern during the whole life of any system, the articles of this session have been rather organised according to a life-cycle criterion. The control system life-cycle (Figure 1) comprises indeed globally two time periods:

- realisation, including the requirements, design, implementation, test and validation phases,
- operation, during which the control system is used to control a physical process or is maintained.

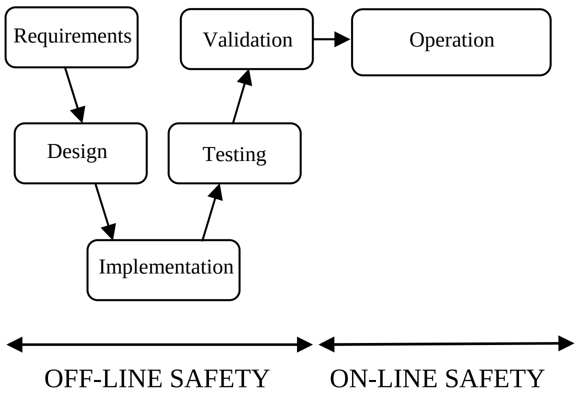

Fig. 1. Control system life-cycle and safety-related methods

System safety attributes can be measured only during operation. Nevertheless they come from realisation activities as well as from operation activities. A hazardous event for instance can be generated by a faulty control program or can result from an operator fault. Two kinds of persons are then concerned by safety: the supplier of the control system and the users of this system. The first one must take care of system safety during the realisation phases, the second ones during operation. By taking into account these two different concerns, the methods contributing to safety can be arranged in two categories: Off-line safety and on-line safety.

The purpose of the *Off-line safety* methods is to minimise the fault risk during realisation, i.e. before the system is used. These methods concern every phase of realisation and provide techniques and tools, such as simulation environments, analysis software, test systems, ensuring the safety of the system under development.

On the opposite side, the objective of the *On-line safety* methods is to ensure safety in an already implemented and running system. One well known method is to establish process supervision using means such as fault detection and fault diagnosis. Supervision enables the operator to spot a fault or a potential source of failure before it has severe consequences. He can then correct the problem with or without the help of a diagnosis software tool. Other systems include reactive checking devices that stop the process as soon as a fault is detected.

Both kinds of approaches: Off-line and On-line are complementary and must be employed. In order to balance the presentations, the *Safe control systems* session gathers together three articles dealing with Off-line safety, and two articles dealing with On-line safety.

## 3. OFF-LINE SAFETY: STATE SPACE SYNTHESIS AND ANALYSIS

The first mean to ensure safety during realisation is to manage the system development as advocated in the life-cycle model. This model indeed provides general guidelines to develop a system in a structured way. This structure enables to better organise the realisation tasks, especially when these tasks are performed by several companies, avoids data losses or changes, and globally contributes to off-line safety in so far as a fault prevention method. Moreover this model addresses verification and validation tasks (Figure 2). In the present context verification and validation shall be understood as defined in Boehm (Boehm, 1979):

- Verification: are we building the product right ?
- Validation: are we building the right product ?

Verification performed during the design and implementation phases aims at checking the intrinsic properties of the models developed during these phases, e.g. stability of a control model, no infinite loop within a software module. Validation is carried out at the end of a phase, when all the developed models have been verified, and consists in checking that the results of this phase are consistent with the results of the previous phase (inputs of the phase under validation), e.g. the behaviour of an implemented software module must comply with its design requirements. The purpose of testing is to check that the implemented system owns all the properties addressed in the design models. At last a global validation enables to check that the properties of the real system delivered to the customer are those required.

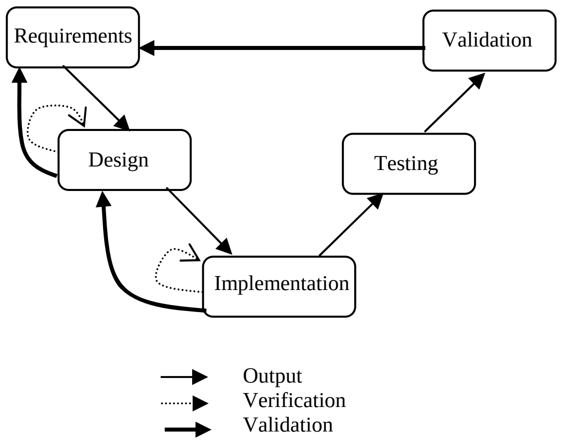

Fig. 2. Verification and validation during realisation

Whatever the interest of the methodological helps provided by the life-cycle model may be, they are not sufficient to guarantee off-line safety. Safe control systems design and implementation require more accurate tools and techniques enabling to minimise fault risk during all design and implementation tasks, by providing for instance helps for control architecture performances assessment, properties checking on control models or software modules. The development of these tools and techniques requires to specify the specific models that they use as well as the possible operations on these models, e.g. a functional architecture definition software manipulates IDEF0 models and helps for operations such as refinement, activities connection. As already mentioned in the first part of this paper, we will subsequently only consider DES control. So all the models generated during design and implementation can be represented as discrete state spaces. Our objective is now to show that off-line safety methods are based upon state space analysis and synthesis.

The aim of design and implementation is to generate a concrete system from requirements. In almost all cases, design starts from an informal specification of the control problem. The term informal refers here to everything that is not based on a strictly composed, syntactically and semantically well-defined form. This informal specification can include for instance “easy to understand” verbal description, timing diagrams, sketches, piping and instrumentation diagrams, or some description of the process to be controlled. In order to ensure safety from the beginning of design, it should also include safety requirements. The designed and implemented system must obviously respect the safety constraints addressed in these requirements.

The first step towards off-line safety consists of the translation of this informal specification into a more formal model, described for instance in temporal logic, by algebraic equations or state automata. Requirements formalisation is not an easy task because the users of the system do not have usually any formal background. Moreover requirements are often confused and contain a huge amount of different kinds of data. However off-line safety requires absolutely to formally express the dynamic properties to be held. If possible, the formal description of the behaviour of the uncontrolled process can also be elaborated. From these two formal models it is possible a priori to automatically generate the formal specification of the control system (Wonham and Ramadge, 1987). Unfortunately this direct synthesis method can not be applied nowadays to an entire industrial system for leading to too huge automata.

It is the reason why industrial control systems design and implementation are till now performed mainly by humans, even if some basic tasks are automated. So control engineers use their skills and their experience to elaborate models from the requirements, with or without a development method specific to the application field considered or imposed by the customer or by the system supplier. The design models are described with languages like state-charts (Harel, 1987), Petri nets (David and Alla, 1992), the implementation models in any controller programming language (synchronous languages, IEC 61131-3 standard languages, for instance). If the syntax and the semantics of the languages are well defined, each of these models can be easily translated into a state automaton, which points out in a formal way the dynamic behaviour of the model.

The power of expression of these languages enables to generate control models for industrial systems. The drawback of this rich potentiality is that the state spaces associated to these models are not controlled by the designer or by the developer. So each elaborated model leads unfortunately to a state space different from that required. This difference can come from a design or implementation fault, i.e. designers have missed or misunderstood some parts of the requirements or, on the opposite side, have been too exacting or have added functions or components that were not required. But, even if no human fault occurs, the use of languages for DES control induces that kind of problem. By noting:

- $\Sigma$ the state space obtained from any design or implementation model, generated in an uncontrolled way,
- $\Sigma'$ the state space implicitly embedded in the requirements and representing the expected behaviour,

the off-line safety problem for DES control systems can then be expressed in the following way:

- The state space $\Sigma$ must include totally the state space $\Sigma'$
- The state space $\Sigma$ shall not contain states leading to hazardous consequences.

The figure 3 shows an example in which the state space $\Sigma$ includes all the required state space (first condition fulfilled), and comprises not required states. Among these last ones, some (on the right side of the line C) do not induce safety problems. They correspond to additional functions. The other ones (on the left side of the line C) are prohibited with regards to the safety requirements. The second condition is not fulfilled.

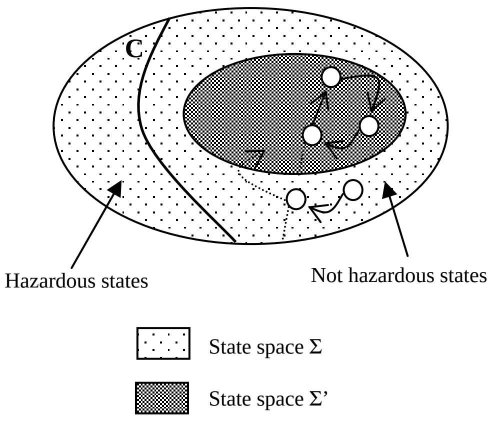

Fig. 3. Required and modelled state spaces

The aim of off-line safety methods is to make sure that the two previous conditions are fulfilled. The main problems when developing such a method arise from the nature of the modelling languages and from the size of the state spaces. Indeed state spaces are not models commonly used during the design and the implementation of an industrial control system and the state spaces $\Sigma$ and $\Sigma'$ are too large to be manually manipulated. Then these methods shall provide software tools for automatic synthesis and analysis of the considered state spaces.

Referring to the figure 3, two approaches are possible. The first one aims at increasing the size of $\Sigma$ in comparison with $\Sigma'$. Use of a generic design model, imposed structure for control models, as it is frequently the case in large companies, are examples of methods based on this approach. The second approach aims at reducing the gap between the two state spaces. This can be done during the elaboration of the control model (a priori checking) or after it has been built (a posteriori checking). The first class gathers the formal methods for automatic synthesis of control systems. Properties checking on an existing model can be performed by simulation or by formal techniques, like model checking (Mac Millan,1993) or theorem proving.

Whatever the approach may be, there is a growing demand for formalisation within off-line safety methods. The presented approach, based on state spaces analysis, provides a general framework to arrange the research works in the domain. The figure 4, adapted from (Frey and Litz, 2000), details some of these works.

In this figure verification and validation shall be understood as stated in (Roussel and Lesage, 1996). Verification is the proof that the internal semantics of a model is correct, whatever its application may be. Properties such as stability, liveness, deadlock are tested during verification. Validation aims at checking if the model complies with the users requirements.

This figure shows that the control model can be synthesised either manually, with may be some methodological help, or automatically. In both cases, a formal model of properties is required so as to obtain a safe control system.

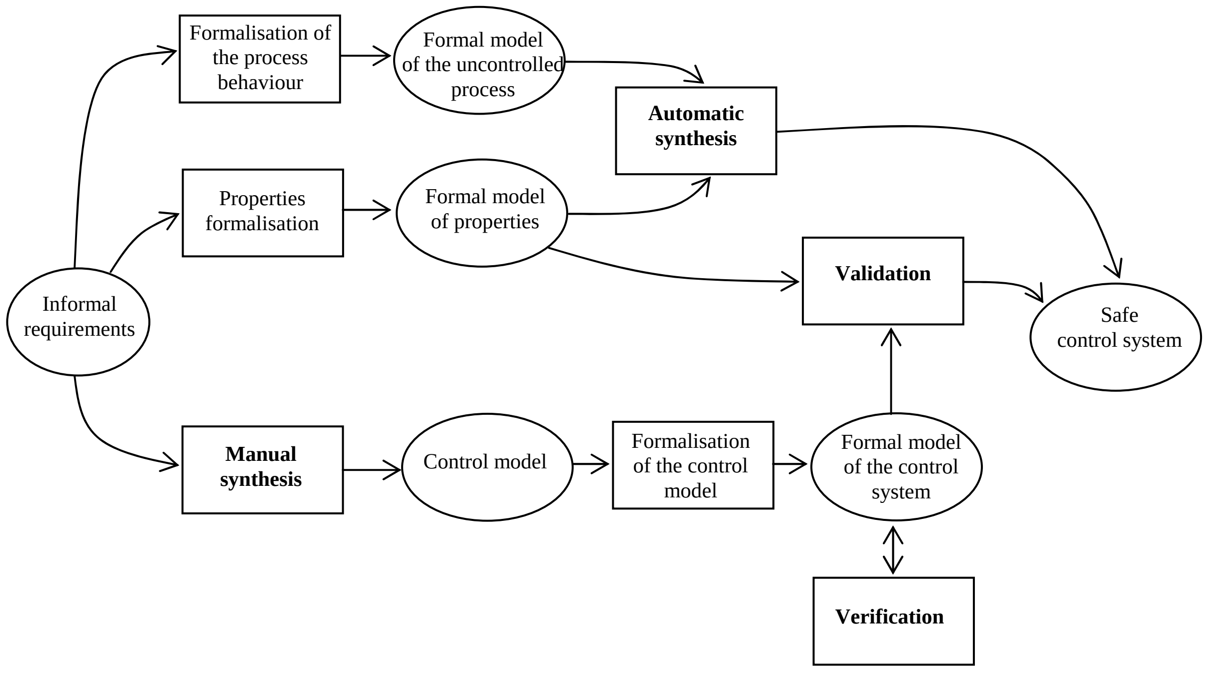
Formalisation of the process behaviour
Formal model of the uncontrolled process
Automatic synthesis
Properties formalisation
Formal model of properties
Validation
Informal requirements
Safe control system
Manual synthesis
Control model
Formalisation of the control model
Formal model of the control system
Verification

Fig. 4. From informal requirements to safe control system

## 4. SESSION STRUCTURE

The *Safe control systems* session gathers papers coming from several French laboratories that belong to a research group entitled Groupement de Recherche en Productique (Research group in Production Engineering). The works of this group are structured according to six issues (for more details see the web site http://www.univ-valenciennes.fr/LGIL/GRP). The authors of this session are mainly concerned by the *Automation of dependable systems* issue.

The papers of this session have been selected so as to give different but complementary viewpoints on the *Safe control systems* issue. The three first papers are related to Off-line safety methods, the latter two to On-line safety methods.

In the first paper, Sahraoui presents a method enabling to build a formal model from a semi-formal specification, a significant concern at the beginning of design. An application case to avionics illustrates the presentation. The aim of the second paper, by Niel, Pietrac and Regimbal is to point out the advantages and the drawbacks of the automatic synthesis method based on the works of Wonham by applying it to a manufacturing process. The third paper, by De Smet, Lesage and Roussel deals with formal verification. Two examples of properties checking on PLC programs by model checking and theorem proving are described and compared. In the fourth paper, Dangoumau and Craye explain how the modes management model for a fault tolerant system can be built. This model is part of the supervision function and enables especially to manage failures. At last, the fifth paper by Vanderhaegen and Millot is not the least, for it is concerned with human-machine systems, a major issue for on-line safety. The results of two studies on cooperative redundancy are discussed.

All the authors hope that this session will show the diversity of the works related to safe control systems as well as the interest of the obtained results, and will induce new efforts in these research fields.

## REFERENCES

Boehm B.W. (1979). Software engineering: R&D trends and defence needs. *Research directions in software technology* (P. Wegner, Ed.). MIT Press, Cambridge.

David R. and H. Alla (1992). *Petri nets and grafcet - Tools for modelling discrete event systems*. Prentice Hall.

Frey G. and L. Litz (2000). Formal methods in PLC programming. *Proceedings of IEEE int. conf. on Systems, Man and Cybernetics, Nashville, USA*, pp. 2431-2436.

Harel D. (1987). Statecharts : a visual formalism for complex systems. *Science of computer programming*. North Holland. **vol. 8**, pp 231-274.

IEC 61508 (1998-2000), *Functional safety of electrical/electronic/programmable electronic safety-related systems. Parts 1-7.* International Electrotechnical Committee.

Laprie, J.C. *et al*. (1996). *Guide de la surete de fonctionnement*. Cepadues.

Mac Millan, K.L. (1993). *Symbolic model checking*. Kluwer Academic.

Roussel, J.M. and J.J. Lesage (1996). Validation and verification of grafcet using state machine. *Proceedings of IMACS-IEEE CESA'96,* Lille, France, July 1996, pp. 758-764.

Wonham, W.M. and P.J.G. Ramadge (1987). On the supremal controllable sub-language of a given language. *SIAM J. Control and optimisation,* **vol 25 (3)**, pp. 637-659.