SAFETY
APPROACHES

IRSN
INSTITUT
DE RADIOPROTECTION
ET DE SÛRETÉ NUCLÉAIRE

Principles relating to the digital instrumentation and control design approach

2017

ABSTRACT

The design of the instrumentation and control of nuclear facilities uses digital systems that offer increasing computation and interconnection capabilities. They enable advanced functions to be carried out, such as calculation of the critical heat flux ratio, help to detect hardware failures in real time and provide operators with rich, flexible interfaces. However, these evolved functions may be affected by faults that make their logic systematically inadequate in certain cases, which introduces sources of failure other than random hardware failures and raises questions about the informal concept of the increased "complexity" of instrumentation and control. Appropriate design principles shall therefore be applied so that this logic is as fault-free as possible and can be assessed by an independent body such as IRSN.

This document presents the main problems associated with the design of the digital instrumentation and control of a complex facility, as well as the general principles to follow to demonstrate that a satisfactory safety level has been achieved. The doctrine elements presented in this document are the result of the experience acquired during assessments carried out for the French nuclear power plants, enhanced by exchanges with experts from the nuclear sector, and reflect French practice; they apply in other sectors in which a high level of confidence can be attributed to instrumentation and control.

The normative texts cited in this document provide detailed requirements that are open to considerable interpretation, as the nature of the problem posed does not enable relevant and measurable criteria to be defined in all cases. This document aims to explain the principles underlying these detailed requirements and to give the means for interpreting them in each situation.

CONTENTS

REFERENCES

[1] Basic Safety Rule II.4.1.a, Logiciel des systèmes électriques classés de sûreté (safety software)

[2] Basic Safety Rule I.3.a, Utilisation du critère de défaillance unique dans les analyses de sûreté (single failure criterion)

[3] Basic Safety Rule IV.2.b, Exigences à prendre en compte dans la conception, la qualification, la mise en œuvre et l'exploitation des matériels électriques appartenant aux systèmes électriques classés de sûreté (electrical systems)

[4] Technical guidelines for the design and construction of the next generation of nuclear power plants with pressurized water reactors, adopted during the GPR/German experts plenary meetings held on October 19th and 26th 2000

[5] ASN Guide n° 22 (Produced jointly with IRSN) : design of Pressurized Water Reactors

[6] RCC-E, Design and construction rules for electrical equipment of PWR nuclear islands (2012)

[7] IAEA Safety Standards, SSG-39 Design of Instrumentation and Control Systems for Nuclear Power Plants

[8] IEC 61226, ed3.0, Nuclear power plants - Instrumentation and control important to safety - Classification of instrumentation and control functions

[9] IEC 61513, ed2.0, Nuclear power plants – Instrumentation and control important to safety – General requirements for systems

[10] IEC 60880, ed2.0, Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions

[11] IEC 62138, Nuclear power plants - Instrumentation and control important for safety - Software aspects for computer-based systems performing category B or C functions

[12] IEC 62566, Nuclear power plants – Instrumentation and control important to safety – Development of HDL-programmed integrated circuits for systems performing category A functions

[13] IEC 62340, Nuclear power plants - Instrumentation and control systems important to safety - Requirements for coping with common cause failure (CCF)

[14] NUREG/IA-0254, Suitability of Fault Modes and Effects Analysis for Regulatory Assurance of Complex Logic in Digital Instrumentation and Control Systems, NRC et IRSN, April 2011

[15] Assessment of the overall Instrumentation and Control architecture of the EPR FA3 project, J. Gassino et P. Régnier, IRSN, EUROSAFE 2010

[16] An experimental evaluation of the assumption of independence in multi-version programming, IEEE Transactions on Software Engineering, January 1986, John C. Knight, Nancy G. Leveson

[17] J. Bickel, Risk Implications of Digital RPS Operating Experience, 2007, AIEA

[18] EPRI, Operating Experience Insights on Common-Cause Failures in Digital Instrumentation and Control Systems, 2008

[19] "Multi-domain comparison of safety standards", ERTS 2010, P. Baufreton, JP. Blanquart, JL. Boulanger, H. Delseny, JC. Derrien, J. Gassino, G. Ladier, E. Ledinot, M. Leeman, P. Quéré, B. Ricque

DEFINITIONS

Instrumentation and control architecture: "*organisational structure of the instrumentation and control systems of the plant which are important to safety*" [extract from reference [9]].

Class of an instrumentation and control system: "*one of () possible assignments () of instrumentation and control systems important to safety resulting from consideration of their requirement to implement functions of different safety importance. An unclassified assignment is made if the I&C system does not implement functions important to safety*" [extract from reference [9], amended to delete the number and names of classes specific to the IEC (see empty brackets)].

Component: "*one of the parts that make up a system. A component may be hardware or software and may be subdivided into other components*" [extract from reference [9]].

Single failure criterion: "*criterion (or a requirement) applied to a system such that it must be capable of performing its safety task in the presence of any single failure*" [extract from reference [9]].

Failure: "*loss of the ability of a structure, system or component to function within acceptance criteria*" [extract from reference [9]].

Single failure: "*loss of capability of a component to perform its intended safety function(s), and any consequential failure(s) which result from it*" [extract from reference [9]].

Common cause failure: "*failure of two or more structures, systems or components due to a single event or cause*" [extract from reference [9]].

Fault: "*defect in a hardware, software or system component*"
"N.B. Faults may be originated from random failures, that result e.g. from hardware degradation due to ageing, and may be systematic faults, e.g. software faults, which result from design errors" [extract from reference [9]].

Systematic fault: "*fault related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors*" [extract from reference [9]].

Instrumentation and control function: "*function to control, operate and/or monitor a defined part of the process*" [extract from reference [9]].

System: "*set of components which interact according to a design, where an element of a system can be another system, called a subsystem*" [extract from reference [9]].

Instrumentation and control system: "*system, based on electrical and/or electronic and/or programmable electronic technology, performing instrumentation and control functions as well as service and monitoring functions related to the operation of the system itself*" [extract from reference [9]].

ABBREVIATIONS

CCF	Common cause failure
CEI	Commission Electrotechnique Internationale (IEC in English)
CHFR	Critical Heat Flux Ratio
IAEA	International Atomic Energy Agency
IEC	International Electrotechnical Commission (CEI in French)
PAS	Process Automation System
PICS	Process Information and Control System
PS	Protection System
RT	Reactor Trip
SAS	Safety Automation System
SICS	Safety Information and Control System

1 BACKGROUND

Instrumentation and control design is an important field in nuclear safety evaluation, as demonstrated by international news in the UK and in Finland, where it has stirred up a great deal of activity within the context of EPR evaluation. This importance also results in sustained activity for international work groups and standardization bodies in this field.

The development of digital technologies enables effective instrumentation and control to be implemented but poses specific difficulties in terms of safety demonstration, which has led interested parties in France (AREVA, EDF, IRSN) to gradually develop a particular approach.
These difficulties emerged suddenly in countries whose nuclear program had not been very active since the 1970s. The situation in France is different in that these techniques were introduced gradually in the P4, P'4 and N4 series, which enabled manufacturers and IRSN to acquire considerable experience, thus creating a more favorable situation. The safety evaluation of the digital instrumentation and control of the Flamanville 3 EPR reactor did, however, raise some delicate issues whose resolution required a strong involvement of the operator, the manufacturers, the Nuclear Safety Authority and IRSN.

This document explains the approach developed in France by presenting the strategy and the principles that IRSN considers have to be implemented when the architecture and the digital instrumentation and control systems are designed (except for the interfaces with the process and human-machine interfaces, for which the document does not go into the specific principles). It applies to nuclear power reactors. Its principles may be relevant for other nuclear facilities but their detailed application will depend on the complexity of the installation and the risks it presents, based on an approach proportionate to the importance of these risks. It lays down the basis for the aforementioned approach and positions it in relation to the international consensus.
It sets out the need for specific design principles to demonstrate that a satisfactory safety level has been achieved. It does not reproduce the detailed requirements which implement these principles as this would result in a document that would be too large and redundant with the normative texts referred to in Chapter 3. However, it exposes the motivations of these requirements and therefore enables them to be interpreted and applied adequately.
It should therefore be treated as the reference base for IRSN's safety assessment rather than an assessment guide.

2 MISSIONS AND ORGANIZATION OF INSTRUMENTATION AND CONTROL

Safety instrumentation and control is involved in monitoring, regulation and protection of the installation. For this, it consists of:

- interfaces with the process: sensors and actuators, either "on/off" or "continuous";
- programmable controllers responsible for acquiring the measurements and commands of the operators, processing them, controlling the actuators and producing the information necessary for operation;
- interfaces with the operators (supervision and control systems) and with the maintenance teams.

Instrumentation and control is organized into systems performing homogeneous functions, of which the following can be distinguished for a nuclear power plant:

- a protection system, which, amongst other functions, performs automatic reactor trip (RT) and start-up of safety systems (for EPR: Protection System, PS);
- a system involved in the functions necessary to achieve the safe state in accident conditions (for EPR: Safety Automation System, SAS);
- a system involved in the automatic and manual functions used in normal conditions, as well as certain limitation functions (for EPR: Process Automation System, PAS);
- a computerized process information and control system, usable (in all situations) as long as it is available (for EPR: Process Information and Control System, PICS);
- a non-computerized safety information and control system for managing incident and accident conditions when the PICS is not available (for EPR: Safety Information and Control System, SICS).

Instrumentation and control systems are organized according to an architecture that aims to satisfy functional requirements (for example, some systems such as the PICS have to communicate with others) and safety requirements (e.g. independence).

3 REGULATORY AND NORMATIVE TEXTS – INTERNATIONAL CONSENSUS

In France, for pressurized water reactors, RFS II.4.1.a [1] deals with the requirements that apply to software, in particular those relating to determinism and predictability that play a key role in the design of safety software (see Chapter 6). RFS I.3.a [2] deals with the use of the single failure criterion and RFS IV.2.b [3] with the requirements that apply to safety-classified electrical equipment. The technical guidelines in reference [4] and the guide in reference [5] also include requirements for instrumentation and control.

Document RCC-E [6] (Design and construction rules for electrical equipment of PWR nuclear islands) produced by AFCEN (French association for the design and construction rules for nuclear steam supply systems) describes the practices adopted by manufacturers for developing safety instrumentation and control in line with the objectives set by the RFSs and the International Atomic Energy Agency (IAEA), as well as the requirements of IEC standards.

The IAEA recommends objectives concerning instrumentation and control. The most recent work is indicated in guide SSG-39 “Design of Instrumentation and Control Systems for Nuclear Power Plants” [7]. Requirements for achieving these objectives are laid down in the standards developed by the committee 45A of the International Electrotechnical Commission (IEC), by virtue of an agreement between these two bodies.

The main IEC standards on safety instrumentation and control are: IEC 61226 [8] (classification of instrumentation and control functions), IEC 61513 [9] (systems), IEC 60880 [10] and IEC 62138 [11] (software), IEC 62566 [12] (programmable electronics) and IEC 62340 [13] (common cause failure).

These essential normative texts in the field of nuclear instrumentation and control are recent and are completely consistent with the approach and the principles described in this document, which therefore are the subject of a broad international consensus.

The doctrine presented in this document explains and justifies the basis for these texts.

4 GENERAL SAFETY APPROACH

The defense-in-depth principle, as referred to in the Decree of February 7, 2012 establishing the general rules relating to basic nuclear installations, is the main basis of nuclear safety. To demonstrate nuclear safety, different levels of defense in depth are defined. Means shall be implemented in order to guarantee the smooth operation of each level, but of which we then assume the failure; other means, sufficiently independent of the previous ones, are defined to limit the consequences of this failure. In particular, requirements relating to the following arise from the defense-in-depth principle:

- prevention of system failures, including common cause failures (CCFs);
- identification of plausible system failures;
- definition of provisions designed to limit the consequences of these failures;
- independence of systems involved at different defense-in-depth levels.

The specific nature of instrumentation and control in terms of technological developments, failure modes and independence, summarized in Chapter 5, led to production of the approach and principles presented in Chapters 6 and 7.

5 SPECIFIC NATURE OF INSTRUMENTATION AND CONTROL

5.1 ADVANTAGES OF DIGITAL TECHNIQUES

In so-called "hardwired" technology, which is used for example for the 900 MWe reactors, the required functionality (such as validation of a reactor trip signal by a permissive) is implemented through physical wiring of discrete components such as relays. The size of these discrete components, in the range of one centimeter, limits how many there are in a system and therefore limits the complexity of the functions that can be implemented.

Digital technology uses integrated circuits that typically contain millions or billions of basic structures[1] equivalent to relays, which gives it a functional capacity that is well illustrated by comparing everyday objects (such as telephones) with their counterparts from the 1960s or 1970s. In the nuclear field, advances, although more modest to retain appropriate control of the systems, are still considerable. They relate, in particular, to:

- **automatic functions**: digital technology authorizes a higher number of functions than the hardwired logic and these functions are more advanced, for example calculation of the critical heat flux ratio (CHFR) in real time using hundreds of sensors and parameters related to design and calibration; it therefore gives a more precise indication of the state of the installation;
- **operation interfaces**: operators have access to a large amount of information and many means of action, the presentation of which can be adapted to the state of the installation and their demand, taking into account requirements associated with ergonomics;

[1] Designers of instrumentation and control do not directly define the wiring of these structures, but develop programs that describe the logic to be implemented in an abstract way, by way of "loops", "variables", etc. Tools translate these programs into interactions of the basic structures contained in "microprocessor" or "programmable electronics" integrated circuits, such as "Field Programmable Gate Arrays" (FPGA).

- **tolerance of operating hazards**: digital architectures enable more maintenance and in-service periodic testing operations to be performed, while detecting and handling abnormal situations such as the simultaneity of maintenance actions on several redundant trains;
- **system availability**: digital safety systems continuously supervise operation of their hardware components, by continually testing the memory cells and the calculation resources, by emitting "life signs" monitored by other systems or sub-systems, etc.; they therefore detect most of their failures, report them and put, in this case, their outputs in predefined positions;
- **diagnosis and maintenance**: thanks to communication networks that enable thousands of pieces of basic information to be transmitted on a single physical medium, the diagnosis units can monitor the self-monitoring reports of the systems in real time and compare many global and intermediate results from redundant entities to detect discordance.

5.2 SPECIFIC NATURE OF FAILURES

Due to the rich functionality mentioned above and the technological complexity that makes it possible, specification or implementation errors may be committed, which leads to permanent faults in the logic of the system in question: in some cases, its response is different from the specified response (implementation error) or the suitable response (specification error).

An instrumentation and control system can therefore be faulty as a result of failures of its hardware components (e.g. a relay or an integrated circuit) or as a result of design faults making its logic unsuitable in some cases. The paragraphs below show that using digital techniques tends to reduce the first causes (or their consequences) and favor the second.

5.2.1 HARDWARE FAILURES

The hardware reliability of the instrumentation and control systems shall satisfy the availability requirements of the functions they perform. It shall be estimated at the design stage, then be monitored during operation to detect any situations that call for corrective measures: indeed, less reliability than anticipated or a reduction in reliability may point to a hardware design error leading to a component to operate outside of its specifications (for example, application of an excessive voltage during switching, inadequate heat dissipation), a component supply problem, a manufacturing defect or even incorrect use of the system.

The quality of the design, of the manufacture and of the operation shall guarantee that each hardware component works within the physical range specified by its manufacturer (voltage, temperature, humidity, vibrations, etc.). Under these conditions, each component has a spontaneous failure probability, in principle known and documented in reliability tables, which the designer uses to calculate the probability of a system failure making it completely or partially unavailable (in the case of systems with redundancy).

However, the reliability tables that are actually available, which are based on experience feedback, do not take include data relating to recent circuits. Some tables can still be used although they have not been updated since 1991. In addition, these tables do not define the failure modes to be taken into account for integrated circuits offering a large number of features. Designers therefore have to reason by analogy and by extrapolation, which lessens the credibility of the values obtained. For old components, it is possible to compensate for this theoretical weakness with favorable experience feedback that demonstrates the conservatism of these estimations.

As indicated in Sub-chapter 5.1, the self-monitoring abilities of the digital systems help to detect and report most instrumentation and control hardware failures, so that only a small fraction of them can result in hidden unavailability of a digital safety system.

Furthermore, some systems important to safety have to accomplish their missions despite a single failure affecting one of their components, while reducing the risk of spurious actions, which calls for redundant architectures. Digital systems permit complex architectures that help to achieve these objectives even in the event of additional unavailability, caused for example by the performance of preventive maintenance or periodic testing during operation.

In an architecture with three redundant trains combined with a 2 out of 3 vote (used for the "hardwired" protection system of 900 MWe reactors), no single failure upstream of the voter can prevent the safety action or trigger it spuriously. Failure of the voter itself can be considered as part of failure of the actuator in question.
An architecture with four redundancies combined with a 2 out of 4 vote (used for the digital protection systems of 1,300 MWe, 1,400 MWe and EPR reactors) tolerates an additional unavailability (for maintenance, periodic testing, etc.) without loss of mission or spurious action. This more complex vote (which needs to be reproduced for each of the hundreds of system outputs) is often done using digital technology. This technology also enables the detection of abnormal situations, such as performance of maintenance work on more than one redundancy, and the handling of these situations in accordance with safety requirements.

In this way, digital technologies improve the tolerance of instrumentation and control systems to hardware failures, at the expense of making architectures and processing more complex.

5.2.2 FAILURES RESULTING FROM AN UNSUITABLE LOGIC

As indicated in Sub-chapter 5.2, failures can occur as a result of a fault affecting the instrumentation and control logic, leading to incorrect behavior in some cases determined perfectly by the characteristics of the fault but unknown until these scenarios are executed (otherwise the fault would be corrected).
Digital technology is more conducive than hardwired technology to faults affecting the logic, because it enables a larger number of more complex functions to be implemented (see Sub-chapter 5.1). This constitutes an important safety challenge for the nuclear sector in that digital technology is now widely used in the safety systems of new installations or during upgrades.

5.2.3 NEED FOR A SPECIFIC APPROACH FOR THE LOGIC

Logic faults in digital systems are not similar to hardware failures and cannot be analyzed, prevented or tolerated with the means suitable for hardware, such as:

- FMEA type analyses (Failure Mode and Effects Analysis): these are used successfully to assess the reliability of a piece of equipment based on failures of its components (relay stuck open, relay stuck closed, etc.) and their propagation paths to the outputs, which follow the physical wiring of the components, itself a reflection of their functional relationships. But the case of logic is different from that of hardware: its faults are unknown (otherwise they would be corrected) and the number of faults that can be postulated is too large to be analyzed completely. Moreover, the propagation paths of each fault that can be postulated are not in principle known. In particular, they are not limited to functional relationships as a fault in one processing can unduly affect a memory used by another processing that is functionally independent of the first. FMEA analyses do not therefore apply to logic faults [14];
- verification by testing at the end of the design stage: testing allows to verify rather simply a hardwired function, which typically has few inputs and internal memories. As the number of possible scenarios increases exponentially with the number of inputs and the number of memories, it is much higher in the case of digital systems and the verification of these systems cannot simply rely on testing at the end of the design stage.

  > 15 analog inputs each digitized over 10 bits (i.e. 2^{10}=1,024 possible values for each one) and 15 internal memories of the same size, i.e. 30x10 input bits in total, can define up to 2^{300}, i.e. approximately 10^{90} different scenarios. To put this into perspective, the universe has approximately 10^{80} atoms. Exhaustive testing is therefore not feasible regardless of the means allocated.

  In this way, the tests cannot be used to verify a digital system with ordinary or unknown design. However, they are a relevant verification means within the context of a design intended for this purpose and enabling, in particular, the possible scenarios to be grouped together into consistent families (see principles in sub-Chapter 6.4);
- diversification: logic faults cannot be easily eliminated, as confirmed by the many failures of everyday industrial products. Furthermore, there is currently no single approach for bypassing their presence and the use of software diversification (see paragraph 6.5.4 and Appendix 3) proves to be ineffective. The addition of diversified backup systems cannot be generalized without making instrumentation and control overly complex, which tends to introduce new faults and to increase the risk of spurious actions (see sub-Chapter 6.5 and Chapter 7); diversification is no substitute for justification of the correct operation of the instrumentation and control systems;
- consideration of a failure rate, which proves inadequate for taking into account the presence of faults (see Appendix 4).

A specific safety approach is therefore necessary for building and demonstrating the correctness of the logic of an instrumentation and control system.

5.3 SPECIFIC ASPECTS CONCERNING INDEPENDENCE

Instrumentation and control is subject to functional requirements that may contradict the need for independence between safety classes and between levels of defense. For example:

- limitation of spurious actions requires a vote based on information produced in redundant trains; communication between "independent" trains is therefore necessary;
- to simplify the design of fluid systems, different instrumentation and control functions with different safety classes or belonging to different levels of defense have to control the same actuator; this dependence between different classes or between different levels of defense is therefore unavoidable in instrumentation and control. For example, in some reactors, the auxiliary feedwater supply of the steam generators is used by operators in normal operation, in the hot shutdown state, but also by the protection system in the event of an incident;
- for ergonomic reasons, operators should use the same computerized means (PICS) in all situations. This control system therefore has to communicate with the instrumentation and control systems used in different situations, which results in links between systems with different safety classes or involved at different levels of defense.

The independence requirement between safety classes and between levels of defense has therefore to be laid down on a case-by-case basis depending on the type of independence required: functional (e.g. no exchange of data or data from a common source), hardware (no common hardware components, such as a processor), physical (e.g. no electrical connection) or logical (e.g. no connection to a same network, even if the computers in question do not exchange information through it).

6 DESIGN APPROACH AND PRINCIPLES

6.1 GENERAL APPROACH TO INSTRUMENTATION AND CONTROL DESIGN

The specific aspects given above call for a specific approach to be adopted, in particular to prevent failures at every stage in the instrumentation and control development process and to take into account residual failures, especially common cause failures, in order to limit their consequences. This approach relies on the definition of and compliance with:

- instrumentation and control specification principles (see Sub-chapter 6.2);
- instrumentation and control architecture design principles (see Sub-chapter 6.3);
- instrumentation and control system design principles (see Sub-chapter 6.4):
    - a strict engineering process and technical principles shall be implemented in order **to avoid faults being introduced** throughout the life cycle of each instrumentation and control system (see paragraph 6.4.1),

- this avoidance approach shall be supplemented by a methodical approach to **remove**[2] **faults**, including non-formal procedures (such as inspections, proofreading, audits and reviews), formal procedures (mathematical reasoning aimed at proving compliance with expected properties, absence of predefined faults, equivalence with a model deemed to be correct, etc.) and validation testing for which coverage shall be justified (see paragraph 6.4.2),
- in addition, the system shall **be able to tolerate some residual faults** that could remain despite the foregoing (see paragraph 6.4.3).

The tolerance of residual faults at system level shall be supplemented by instrumentation and control architecture measures including diversification (see Sub-chapter 6.5), in the knowledge that these faults can also result in spurious actions, which have to be analyzed (see Chapter 7).

Lastly, all of the instrumentation and control development phases shall also take into account protection against malicious acts (see Sub-chapter 6.6).

In the case of the Flamanville 3 EPR reactor, fundamental choices were made from the preliminary instrumentation and control design stages in order to take into account the principles arising from the teachings of previous designs and their evaluations by IRSN:

- the elimination of certain functions of the diagnostic tools, whose interaction with the SAS (Safety Automation System) could have prevented demonstration of compliance with this system's response time requirements;
- the introduction of a network within the SAS to be able to demonstrate that communication between its units is always done within a known time, an essential condition for demonstrating compliance with the global response time requirements of the SAS;
- the overhaul of the architecture of the engineering tools, to be able to demonstrate that they do not excessively influence operation of the SAS and of the Process Automation System (PAS) and therefore eliminate potential CCFs caused by this common factor;
- the limitation of the transfer rate of certain computers on the networks, in order to eliminate CCFs caused by potential “jabbering” of a computer on a network, which would prevent other computers accessing it;
- the introduction of a hardware button for validating permissive orders issued by the PICS (Process Information and Control System, computerized) to the PS, to prevent propagation of PICS failures to the PS;
- the creation of a monitoring means for detecting PICS failures and notifying operators of the need to switch to the SICS (Safety Information and Control System, not computerized), with a confidence in keeping with the classification of the latter.

[2] The word “eliminate” could also be used in the current document: it is often used in the context of I&C with a less strong meaning than in the general context of nuclear safety. In the French version of the document, another word is used.

6.2 INSTRUMENTATION AND CONTROL SPECIFICATION

Instrumentation and control shall be developed from written requirements which have to be complete (each important aspect shall be dealt with), clear (understandable to somebody with general knowledge of the field but not necessarily knowledge of the specific project) and precise (unambiguous). In particular, they shall cover:

- the detailed description of each function to be implemented, including the precision, response time, downgraded modes (for example, calculation of the CHFR with one or more sensors detected as faulty), the fallback positions in the event of a failure being detected, etc.;
- the interfaces between functions;
- the principles of priority between functions acting on the same actuator or on actuators that are functionally connected;
- the safety classes of the functions, which lead to design, manufacturing and operation requirements for the systems that implement them;
- the independence of the functions, for example when they are involved at different levels of defense;
- the constraints imposed by the installation design, for example the interfaces with the electromechanical equipment and the support systems, the ranges of the signals, the location of the equipment and the routing of the cables;
- the operating and maintenance constraints;
- the environmental conditions and the hazards to be considered, as well as the corresponding qualification requirements.

These requirements shall be consistent with those taken from the nuclear safety studies.

6.3 ARCHITECTURE DESIGN

Instrumentation and control shall be organized into systems, ensuring in particular to:

- define systems that are consistent from a functional point of view, in order to limit communication between systems;
- check that the number of redundant systems or sub-systems satisfies the single failure criterion where it applies;
- comply with the requirements resulting from the safety class of each function, as well as the separation and independence requirements of the functions involved in different levels of defense;
- transmit signals between systems and conduct votes and priorities between their outputs in accordance with separation and independence requirements;
- organize interactions between men and machines (control systems, production of alarms, diagnostic and maintenance tools, etc.);
- use failure signalization to notify operators of any system unavailability and provide the means to switch over, automatically or manually, to backup systems;
- specify the requirements of the different systems, their interfaces and the communication between them, so that their combined operation satisfies the requirements regarding processing to be done, precision, response time, etc. (see Sub-chapter 6.2).

The architecture shall be designed according to a planned engineering approach, paying special attention to verification activities. The requirements imposed on each system shall include use of an appropriate engineering

process and, depending on its safety class, use of techniques that favour correct operation and verification (see Sub-chapter 6.4).

An ideal architecture would avoid any communication between different levels of defense and between different safety classes, and would use diversified solutions for the different levels of defense. However, this ideal is not achievable because of the functional constraints imposed on instrumentation and control (e.g. those presented in Sub-chapter 5.3) and the limited opportunity for technological diversification. There are only very few industrial solutions that can actually be safety-classified; in addition, having more technologies (and therefore tools and design and operation procedures, as well as interfaces with users) makes design and operation more complex, which can affect safety (see paragraph 6.5.2 and Appendix 2).
The designer shall therefore make choices and justify them (see [15]), particularly in relation to the objective of preventing CCFs between redundant systems or sub-systems, or between systems implementing independent functions.

CCFs caused by propagation of a failure or inadequate interaction between systems shall be prevented, from a hardware point of view, by the physical and electrical separation of the entities in question and of their support systems (in particular their power supplies). This means electrically decoupling communications, for example with fiber optic links. From a logical point of view, each interaction shall be analyzed from the following angles:

- functional, by analyzing the role of the data exchanged, which also concerns the fields upstream of instrumentation and control;

  Analysis of the functional role of each piece of data received by a system allows to determine the possible consequences of this data having an incorrect value, caused by the failure of the issuing entity or the transmission. If these consequences are unacceptable, the designer typically has to resort to redundancies and votes to guarantee a correct result.

- technological, by analyzing the communication protocols and their influence on operation of the computers. The need to master this influence places constraints on system design (see Sub-chapter 6.4 and Appendix 1) and can prevent the use of standard “programmable logic controllers”. Indeed, the design of these controllers tends to generalize interconnections to increase flexibility, which complicates the analysis and can even make it impossible.

In typical industrial systems, when several units are connected to a same network failure of a unit can disrupt others even if they do not exchange information. For example, if a faulty unit is permanently "jabbering" on the network, the others can no longer transmit or receive information through this network. In safety systems, the designer have therefore to use more networks, separate from one another, to limit the consequences of a failure.

Between two communicating units, standard communication protocols can result in blocking of a healthy unit if it is waiting indefinitely for information that will never arrive because the transmitting unit is faulty. Typically, in quadruple redundant protection system, four units calculate an automatic scram condition and a voter receives the information from the four units through four separate networks. This architecture is reliable, but the voter shall not continuously wait for information from the four units and get blocked if one of the four pieces of information does not arrive. Safety systems therefore need specific communication protocols, guaranteeing that failures will not be propagated.

CCFs caused by the coincidence of failures shall be prevented, with regard to hardware, by its reliability, periodic testing of frequency consistent with its failure probability, and its ability to operate in the specified environmental conditions. With regard to the logic, there are no recognized means for evaluating or even properly defining the probability of faults being present (see Appendix 4): this aspect can therefore only be handled qualitatively. The design of the systems shall thus ensure that a residual fault potentially common to several entities cannot be activated simultaneously in them (see Sub-chapter 6.4); concerning the architecture, the designer shall identify the residual plausible causes of CCFs, postulate their existence and limit their consequences by means of diversified functions or diversified systems (see Sub-chapter 6.5). Identification of the residual causes of CCFs affecting several systems shall take into account their operating and communication principles (e.g. cyclical or upon request), the languages and software tools used and their main hardware and software components (processors, network interfaces, libraries, etc.).

A commissioning plan shall ensure the correct on-site integration of the instrumentation and control systems validated separately in the manufacturer facility. This plan shall justify the ability of the planned tests to exercise the systems sufficiently to confirm that they operate correctly on site and interact, with one another and with the installation, in accordance with the specified requirements.

6.4 DESIGN OF A SYSTEM

6.4.1 AVOIDANCE OF FAULTS

The design shall avoid introducing faults, in accordance with the following principles (detailed in the documents referred to in Chapter 3):

- organization of system development into clearly identified phases, with input and output documents defined beforehand, subjected to systematic reviews and verifications; there shall be enough phases to limit the extent of the activities carried out in each one, in order to reduce the risks of error and to facilitate verification (see Figure 1 below);
- systematic identification of requirements (functional and non-functional) and verification of their consistency with those of the installation;
- design rules to structure the logic, exclude the constructions conducive to the faults and facilitate verification by manual or automatic analyses;

- systematic documentation and justification of the design choices;
- management of the configurations of the technical products and of the documentation of the activities, to ensure their availability and the control of their changes.

Figure 1 illustrates the typical "V-cycle" organization of the phases; the design phase is in fact divided into as many sub-phases as required to limit the extent of the activities performed in each. These activities aim to divide the initial problem into sub-problems that are simple enough, allocated to hardware or software components.

These components are then implemented (at the bottom of the V-cycle): this involves, for example, the programming of the software components. In the upward branch of the V-cycle, the components are assembled gradually ("integration") by ensuring that they interact as intended by the design. The order in which the hardware and software components are integrated generally depends on their interactions and therefore on the design.

All of the preceding phases are verified, as shown in Figure 1, and the fully integrated system undergoes a validation (see paragraph 6.4.2).

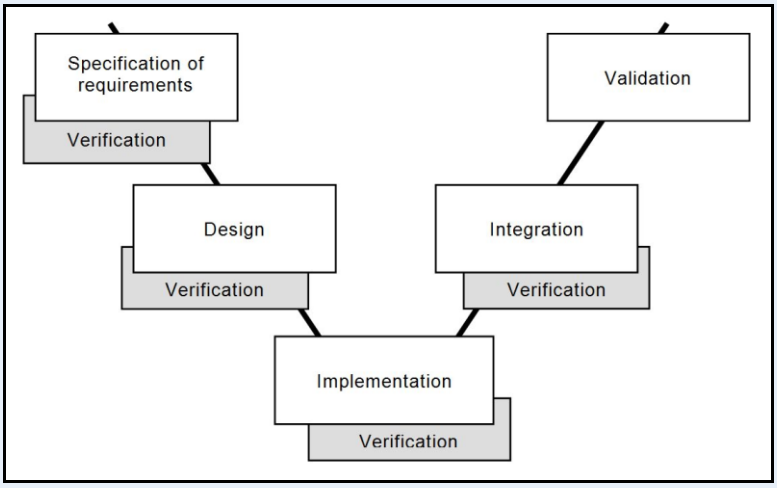

Figure 1: principle of a "V" development cycle according to the IEC

Changes can be made, for example in the event of the functional requirements changing or an anomaly being discovered. Any changes shall be made in accordance with a procedure consisting of formalization of the request, evaluation of its relevance and analysis of its impact on the documents, the software, the rest of the system and its performance. If the request is accepted, the development process shall start from the first phase affected by the change, in accordance with the same design, verification, documentation and configuration management requirements as for the initial development. The amended documents shall refer to the change request and a document shall summarize the actions performed.

The principles concerning the process are supplemented by others relating to the product itself. For systems classified at the highest safety level (1E in the case of the N4 series, F1A in the case of the EPR reactor), the designer shall, in particular, demonstrate that, in accordance with RFS II.4.1.a [1], each computer operates deterministically (the calculation cycle and the use of resources, such as the memory, the networks, the inputs and the outputs, have to be determined at the design stage) and its results do not depend on factors other than the inputs specified. In practice, this principle strongly constrains the design and makes the verification more effective.

In the field of office automation, the behavior of a given software program often depends on an internal database, which is continuously modified by other software and by user actions, which makes each computer unique. Therefore, the behavior of such a software program depends on an unstable environment, which drastically reduces the significance of the tests (as the same action can have different consequences, depending on a context that cannot be controlled) and finally causes most of the malfunctions encountered in this field.
By contrast, determinism involves always doing the same calculations, based on the same clearly specified input data and using exclusively the resources specified and allocated during the design; this property enables the software to be analyzed independently of its context and representative tests to be produced.

In addition, the following principles concern more specifically the avoidance of CCFs (they are presented more precisely in Appendix 1 and detailed in standards [10] and [13]):

- failure of a computer (resulting from a residual fault or a hardware failure) shall not propagate to redundant or independent computers, even if it communicates with them; to do this, they shall be mutually asynchronous, without ever waiting for an action from another for a time greater than the limit taken into consideration in the demonstration of determinism;

  Under this condition, the data received by a computer can be incorrect or absent in the event of failure of the transmitter, but its calculation cycle is not affected. It can therefore handle this scenario in a functionally correct manner, for example using redundant input data and votes.

- no operating condition can simultaneously activate a potential fault in several redundant or independent computers; in particular, the sequence of operations and the resources needed shall not depend either on the process status or transients, or on explicit or implicit information that may be common to several computers, such as the calendar date (“Millennium Bug”) or the time since their restart.

For example, if the time since the restart was involved in a non-functional processing (relating to self-monitoring, maintenance, etc.) a fault (such as the unanticipated "overflow" of a time counter after a certain time) could cause the CCF of independent computers performing different instrumentation and control functions.
Similarly, if the network load or the needed calculation resources varied depending on the transient conditions of the process, a CCF could occur during unexpected transient conditions.

RFS II.4.1.a [1] requires that systems classified at the second safety level (2E in the case of the N4 series, F1B in the case of the EPR reactor) are "predictable"; this principle gives the designer more freedom than the determinism principle presented above, by enabling variations in the calculation cycle or the use of resources (such as the memory, the networks, the inputs and the outputs), as long as the results comply with the functional requirements throughout the range of these variations. In practice, this requires the design to be mastered enough to be able to determine this range.

All of the factors (functional inputs and resources) on which a calculation depends can be identified: compliance with the determinism principle can therefore be verified. However, determinism (and to a lesser extent "predictability") significantly constrain the design, lead to resources being underused and call for major justification work, in such a way that they are only required in sectors governed by very strict technical regulations, such as nuclear power and avionics.
From the assessor's point of view, this means that products accepted in other industries are not acceptable without an additional demonstration for safety-classified systems: they are, until proven otherwise (and in fact often), based on a design that is neither deterministic nor predictable, and thus cannot be verified. Indeed, the very many possible scenarios for even a simple logic (see paragraph 5.2.3) weakens empirical evidence based on experience feedback: even by observing millions of copies of a system for millions of years, the proportion of possible scenarios covered would still be very low and its representativeness for a different use would be uncertain[3], for lack of adequate control of the design. This confirms the need for a specific approach for designing nuclear safety instrumentation and control systems.

The design principles above are broken down into more detailed requirements in normative texts such as [7], [10], [11], [12] and [13] (see Appendix 1). These texts also give information at the even finer level of implementation techniques, which may be appropriate in some cases but not in others, depending on details that are beyond the scope of this document (see, by way of an example, the use of "interrupts" in software in Appendix 5).

[3] The failure of the Ariane 501 launcher is typical: its inertial reference system, developed for Ariane 4 based on good industrial practice, worked perfectly in this launcher despite the presence of a fault that did not reveal itself in this context; the different velocity profile of Ariane 5 indirectly but deterministically resulted in the activation of the fault and the failure of the launch.

6.4.2 ELIMINATION OF FAULTS

Faults that have, despite everything, been introduced into software shall be searched for and removed in accordance with the following principles:

- for systems classified at the highest safety level (1E in the case of the N4 series, F1A in the case of the EPR reactor), the products of each software development phase shall be verified by a team independent from the one responsible for the design; for the other safety classes, the main products shall be verified by people who were not involved in their creation (crosschecks within a single team are therefore permitted in this case);
- the activities, means and objectives of the verification shall be justified in advance; the verification shall be based on a consistent set of analyses of documents, tests and software analyses; technical reviews shall be undertaken to assess the completed design and verification activities and come to a formal conclusion regarding their acceptance;
- the final product shall be validated by tests demonstrating that it operates in accordance with its requirements; for the highest safety class, these tests shall be built by the independent verification team and their ability to extensively exercise the final product shall be demonstrated.

Testing strategy

Justification of the relevance of the tests is a tricky subject Indeed, safety software is affected by few faults, unlike standard software for which tests defined without a rigorous method allow to detect faults. The relevance of the safety software tests cannot therefore be justified by the number of faults detected, but only by an analysis of their ability to detect potential faults.

This justification needs the requirements of the software to define a limited number of families of execution scenarios (e.g. the requirement “if the pressure is greater than 150 bar, trigger the automatic scram” defines the two families “pressure greater than 150 bar” and “pressure not greater than 150 bar”) and needs the design to guarantee the uniqueness of the processing for all the scenarios of a given family; otherwise, the scenarios cannot be grouped into consistent families and verification is impossible.

A limited number of test cases, for example one scenario per family and scenarios on the borders between families, can therefore be representative enough: for the previous example, scenarios with pressure values 75, 149, 150, 151 and 160 bar could therefore be chosen. The tests shall be built by the verification team and the expected outputs shall be determined before the tests are carried out.

Under these conditions, the performance of the tests with actual outputs complying with the expected ones confirms that the families of scenarios specified by the requirements are correctly implemented. If the outputs do not comply with the expected ones, an analysis shall be conducted to identify all of the causes of the fault, including in the development process, in order to eliminate them.

However, the execution could include processing not specified by the requirements, which may produce incorrect outputs. For example, the implementation could use in the calculations the absolute value of an input instead of the input itself; the output would therefore be incorrect when this input is negative, and the tests could not

include any scenarios of this family as it is not specified by the requirements. Therefore, these tests would not necessarily detect the fault.
To deal with this eventuality, the verification team shall analyze the "structural coverage" of the tests, i.e. their ability to exercise the elements of the internal structure of the software in question, such as the processing or the variables. In the example above, this analysis would detect the presence of absolute value processing, such as "change the sign if the input is negative", not exercised by the tests as these do not include scenarios with a negative input value. The rest of the analysis would show that the presence of this processing constitutes a fault. In other situations, the analysis could show that the processing not exercised is specified by the requirements but that the tests are incomplete, which would result in the test building method being reexamined.
Thus, the analysis of the ability of the tests to exercise the structure of the software in question enables, if they are built without knowing this structure, to confirm the absence of processing not specified by the requirements[4].

The double confirmation of correct behaviour of the specified processing and the absence of processing not specified constitutes an essential result for the verification, on the condition that the tests are based on the requirements, before their structural coverage is analyzed. The tests shall therefore not be built with knowledge of the structure of the software in question and aim for example to exercise the processing actually implemented: this approach ("structural testing") is often adopted in industry as it requires little effort, but it is not acceptable for safety-classified systems as it comes down to checking a product against itself and not against its requirements.
Analysis of the structural coverage means deciding what types of structural elements shall be called upon by the tests performed beforehand; indeed, the types (instructions, conditional processing, processing between writing and reading of each piece of data, etc.) relevant for this analysis depend on the design and programming choices made for the software in question and cannot therefore be identified once and for all. The types chosen for the software in question shall therefore be documented and justified.

Compliance with the above-mentioned conditions requires great care but, under this condition, the tests are a relevant and essential verification method. Standard [10] details the principles described in this paragraph and contains an appendix that illustrates the scopes of different test techniques.

Formal verification

Another verification approach that is currently being developed relies on the observation that the logic of a digital system is a set of equations between the inputs and the outputs, and therefore a mathematical object; as for a theorem, its properties are primarily proved by reasoning on groups of scenarios and not by trying to examine them individually. This observation is one of the bases for the formal verification, which involves analyzing a program, as automatically as possible, to prove that it has certain properties, for example that a certain

[4] The presence of processing not specified by the requirements can however be justified, for example in the case of preexisting libraries being used. In this way, a software program that needs the "sine" function can use a library of trigonometric functions including other functions such as "tangent": the complete software thus includes functions not specified by the requirements, which may be acceptable if the included library had been properly developed and verified.

relationship between inputs and outputs is always true or that a variable that constitutes the denominator of a division is never zero, regardless of the program inputs.

Despite certain theoretical[5] and practical[6] limitations, this approach has already been used successfully by EDF for the protection systems of the EPR reactor and the 1,300 MWe reactor series during the upgrade associated with their third ten-yearly in-service inspection. Its implementation shall be clearly specified so that its scope and its complementarity with the other verification methods can be evaluated.

6.4.3 TOLERANCE OF A SYSTEM TO RESIDUAL FAULTS

Despite all of the precautions taken to avoid logic faults and to make the hardware reliable, failures may occur. Insofar as is possible, they shall be taken into account in the system in question by:

- redundancies in light of the hardware failures;
- self-monitoring mechanisms detecting abnormal behavior (see Sub-chapter 5.1) without making the design overly complex, which implies not seeking to identify the cause of the anomaly in the safety system itself; this search can be performed by an external diagnostic system, receiving information from the safety system without transmitting any to it;
- the prior definition, functionally justified, of values that the system's outputs have to take if abnormal behavior is detected by the self-monitoring mechanisms.

6.5 ARCHITECTURE LEVEL CONSIDERATION OF THE POSTULATED FAILURES - DIVERSIFICATION

6.5.1 GENERAL PRINCIPLES

Despite compliance with the principles outlined in Sub-chapter 6.4 to avoid system failures, such failures shall be postulated and instrumentation and control architecture provisions shall take them into account, limiting themselves to plausible failures to avoid making things overly complex: for example, the CCF of all of the digital safety systems is not plausible solely because of them being digital in nature if they do not exchange information and use different types of computers complying with the design principles that allow demonstrating that their behavior is correct and independent of each other and of the environment. Now, the designer shall postulate:

- the existence of a fault in the logic performing an instrumentation and control function, because of an incorrect functional requirement or a design error; this fault could make this function fail simultaneously in the redundant trains;
- the existence of a fault in the logic performing the basic functions of a type of computer (e.g. PS computers), resulting in a CCF of several computers of this type (although the mechanisms that can

[5] For example, the undecidability of certain properties, in the logical sense (within the mathematical theory, the property cannot be proved nor disproved) or in the algorithmic sense (there is no program capable of doing the demonstration in a finite time).

[6] For example, the size of the equations that represent a given software. Safety software designed in accordance with the principles of this document facilitates the use of formal methods, as the relative simplicity of their functional requirements and the consistency of their design limit the size of these equations.

simultaneously trigger or propagate failures have been removed by application of the design principles in paragraph 6.4.1).

The tolerance to these postulated faults relies on the addition of adequately diversified backup systems.

6.5.2 IMPACT OF DIVERSIFICATION ON ARCHITECTURE COMPLEXITY

As potential faults in the logic of an instrumentation and control system result mainly from excessive complexity, the diversification approach shall avoid an excess of backup systems, the presence of which could provide reassurance in a simplified representation of instrumentation and control, but which would in reality make it more complex, at the risk of introducing faults into the architecture or into systems that were correct until then. Thus:

- a greater quantity of hardware and more types of hardware create more maintenance difficulties; and yet, in practice, maintenance errors are the main source of CCFs of digital safety systems, way ahead of logic faults (see Appendix 2);
- the addition of different interfaces with the operators, arising from the addition of different systems, can increase the risk of in-operation errors, especially in difficult situations;
- an actuator controlled by several systems (the normal system and the backup system) can receive contradictory commands in the event of a failure; this conflict shall be resolved by the actuator's control logic, which is a single point of failure of the architecture; this increases the complexity of this logic, at the risk of introducing faults.

All of the consequences of adding systems shall therefore be examined in order to identify a solution that is satisfactory from a safety point of view.

6.5.3 APPLICATION FOR THE FLAMANVILLE 3 EPR REACTOR

Within this context, the designer of the Flamanville 3 EPR reactor introduced various diversifications (see Figure 2 below).

Functional diversification within the PS helps to overcome a fault that would affect a protection function by way of other functions, enabling the same objectives to be met by means of different physical signals and different processing. For example, the function "Reactor trip on steam generator low level" is diversified by the function "Reactor trip on low CHFR". This diversification is effective as these functions are independent despite the fact they are installed in the same system. This independence relies on the determinism of the operation (see paragraph 6.4.1): the results of each function only depend on the resources allocated to it and on its specified inputs, and the diversified functions do not have any resources or specified inputs in common; they are therefore independent. The demonstration of separation of the resources attributed to the diversified functions has been facilitated by implementing them in separate computers.

Technological diversification helps to overcome a common cause failure of the PS computers (implemented in Teleperm XS technology) in the situations whose importance has in particular been highlighted by the probabilistic safety studies; it consists in protection functions in the event of ATWS (Anticipated Transient Without Scram), implemented in the SAS which uses a different technology (SPPA-T2000) than that of the PS with regard to the operating mechanisms, the hardware and software components, the languages, the network protocols, etc.

Another technological diversification helps to overcome a common cause failure of the SAS computers: in a symmetrical way to the previous one, the functions of the CCND (hard-kernel instrumentation and control, implemented in Teleperm XS technology) diversify certain functions of the SAS implemented in SPPA-T2000 technology.

A third technological diversification helps to overcome a common cause failure of the PICS computers (computerized) by the SICS (not computerized).

Direct connections between the control room and certain sensors and actuators complete these diversifications.

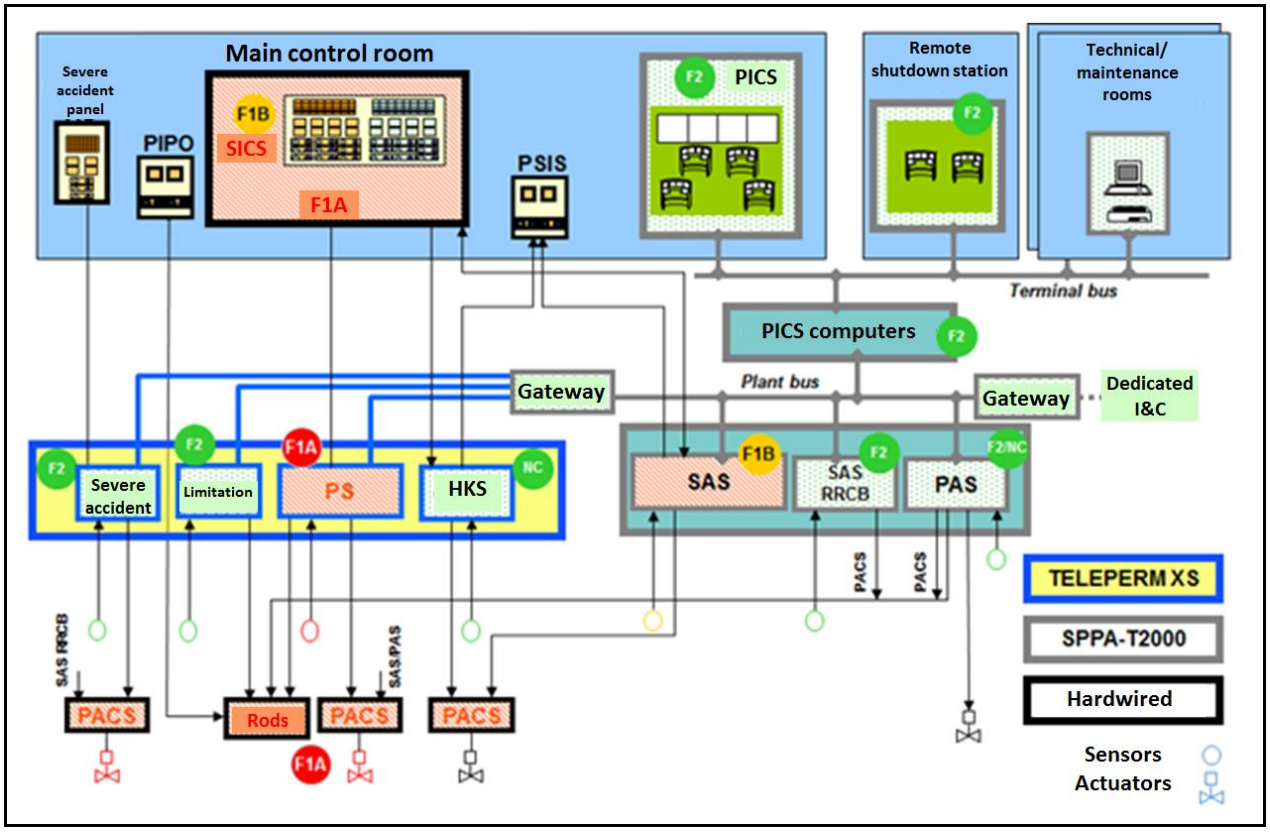

Figure 2: diagram of the instrumentation and control architecture of the Flamanville 3 EPR reactor

6.5.4 INEFFECTIVENESS OF SOFTWARE DIVERSIFICATION

The diversification of a software program has been proposed as a way to overcome the consequences of logic faults. This approach involves executing in parallel several programs ("versions") that perform the same function but that were developed independently, based on the assumption that the faults they may have do not concern the same scenarios and can therefore be masked by an appropriate vote.

Knight and Leveson [16] tested out this approach with 27 versions of the same function, under conditions favorable to diversification (see Appendix 3). And yet the number of scenarios in which several versions simultaneously provided incorrect results proved incompatible with the assumption of statistical independence between faults that motivated the approach, leading with 99% confidence to rejection of this assumption.

The diversification of a software program cannot therefore, without further evidence, be considered as a measure to counter the presence of hypothetical faults.

Thus, a generalized software diversification requirement could adversely affect the quality of each version of this software by diluting the means, would require a logic to elaborate a global result that itself could be affected by faults and, more generally, would make instrumentation and control more complex in the hope of an improvement that is not confirmed by experience.

6.6 CONSIDERATION OF MALICIOUS ACTS

Some safety system design principles presented in this document favor the protection against malicious acts, for example:

- the principles of simple and justified design, independent verification and rigorous configuration management complicate the introduction and concealment of malicious software;
- trying to avoid faults in the software prevents them being used to carry out attacks designed to cause abnormal operation (execution of unintended programs, access to resources that are normally inaccessible, etc.).

For example, if the user is invited to enter information (name, address, etc.), this is written in a memory area, often of a fixed size (e.g. 500 characters); if the information entered exceeds this size and if the program does not protect itself against this scenario (which constitutes a fault), the writing can overflow into the neighboring memory areas; depending on the role of these areas (e.g. designate the processing to be done next), this may result in functions not anticipated by the designer being performed. This type of attack does not need any skills in difficult fields such as cryptography and, although old, is still effective in office automation and standard industrial sectors because their software is affected by many faults.

However, consideration of malicious acts can influence the choice of organizations to be put in place throughout the different development and operation phases, as well as the architectures, components, software and their location, in order to increase the robustness inherent in instrumentation and control. Therefore, in all of its

phases, the design shall take into account protection against malicious acts. In addition to the aforementioned choices, specific measures shall be taken to detect, prevent or at least delay and minimize the consequences of a malicious act and therefore protect the availability and integrity of the instrumentation and control functions.
For example, the design shall prevent access to systems from outside the installation, use technical means to prevent physical access to sensitive equipment by unauthorized persons and, insofar as is possible, prevent inappropriate changes to programs or data.
These specific means necessary to protect against malicious acts can make instrumentation and control more complex and introduce new risks of faults that could affect the functionality of the systems. A balance should therefore be sought between what such means contribute to protection against malicious acts and how much they increase the system's complexity. Consideration of malicious acts from the beginning of the design phase favors the achievement of a satisfactory balance.

7 SPURIOUS OR INAPPROPRIATE INSTRUMENTATION AND CONTROL ACTIONS

The consequences of each single spurious action shall be studied: in practice, such an action is assumed to lead to spurious operation of the controlled actuator, which constitutes a case of active failure as defined by RFS I.3.a [2]; it shall be taken into consideration in system design and in the safety studies.

For the Flamanville 3 EPR reactor, a spurious action of instrumentation and control shall not cause any reference incidents (PCC3) or reference accidents (PCC4); however, it may lead to a reference transient condition (PCC2). To achieve this objective, the actuation of a mechanical component whose spurious operation would result in an operating condition more severe than a transient condition (PCC2) requires a combination of actions from different divisions of instrumentation and control.
Therefore, opening a steam relief isolating valve (“Valve” in Figure 3) requires two control solenoid-valves to be opened: EV1 and EV2, opened by instrumentation and control divisions 1 and 2 respectively, or EV3 and EV4, opened by instrumentation and control divisions 3 and 4 respectively. Thus, a spurious action, or more generally a single failure of the instrumentation and control, cannot trigger the spurious opening of the valve or prevent it opening if needed.

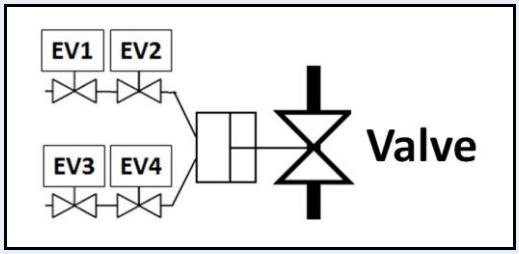

Figure 3: principle of control of a steam relief isolating valve

With the growing use of digital techniques, instrumentation and control architectures tend to group together control of different actuators in the same computer or in computers that communicate with one another, which

may introduce the possibility of multiple spurious common cause actions. It is important to ensure that this cannot lead to a situation that has not been considered by the operating condition studies.

Compliance with the principles presented in this document allows demonstrating that the independence required between functions and between systems is effective and thus considering that a CCF resulting in spurious or inappropriate actions of multiple actuators corresponding to independent functions is implausible. The multiple spurious actions still to be considered are:

- those caused by a single hardware failure (e.g. failure of an electronic board supporting several outputs to actuators), where this is plausible;
- the spurious start-up of a function in an instrumentation and control division (that does not necessarily result in spurious operation of a fluid system, as the example in Figure 3 shows).

The functional consequences of these scenarios shall be analyzed; in particular, the spurious start-up of a safety function can block another safety function and therefore may be unsafe.

For example, start-up and isolation of the emergency feedwater of a given steam generator are two antagonistic functions with have the highest safety classification; this means that the action carried out in the event of simultaneous requests had to be chosen in advance: as it is, isolation takes priority over start-up so as to guarantee isolation of a steam generator in the event of a break in one of its pipes.
The spurious actuation of the isolation function could therefore block the start-up function in a situation that needs the latter to evacuate the residual power. This situation has been studied from a functional point of view, beyond instrumentation and control: it is acceptable because the residual power can be evacuated by the three other stream generators.

The MDEP (Multilateral Design Evaluation Programme) task force of the Nuclear Energy Agency questioned the USA, the UK, China, France, India and Finland about their position with regard to consideration of spurious actions. Their positions are similar to the one expressed above, although sometimes not as detailed: the demonstration shall study the scenario of a plausible spurious action, and the justifications relating to the design (quality resulting from the classification and independence) help to limit the scope of the multiple spurious actions to be considered.

A reactor protection system (for the N4, P4/P'4 series renovated during their third ten-yearly in-service inspection and Flamanville 3) uses several computers, of a given type for a given reactor series; computers of the same type include the same basic software. The utility has demonstrated that these different basic software programs meet the strict requirements of standard [10] and operate independently of their environment (process, calendar dates, elapsed time since restart, etc.). Each copy of a software basic program therefore runs at its own pace, without coupling with an environment that could activate a hypothetical error in the different copies simultaneously.
This means that the scenario does not have to be considered in which different functions, implemented in different computers of a protection system, would be started up spuriously and simultaneously because of a CCF activated by coupling of the basic software common to these computers with the environment.

8 CONCLUSION

Digital techniques enable advanced functions such as calculation of the CHFR to be carried out, help to detect hardware failures in real time and provide operators with rich, flexible interfaces. These functions may, however, be affected by faults that make their logic inadequate in certain cases, which introduces sources of system failure other than random hardware failures and raises questions about their consequences.

Hardware failures affecting the safety systems are taken into account by redundant architectures and by periodic testing of frequency consistent with the estimated frequencies of the hardware failures. But faults affecting the logic are not the same as hardware failures, as they affect the outputs systematically rather than randomly, and cannot be prevented or studied using the same means.

The traditional logic development approach, used for example in office automation, does not adequately control the design and results in products that cannot be verified and that are affected by many faults. Furthermore, the attempts made to control the reliability of a system without prioritizing elimination of the faults in its logic have proven inadequate: for example, the diversification of a software program by the way of several versions running in parallel, in the hope of concealing the faults in each one by a majority vote, is not actually very practicable and its effectiveness has been contradicted experimentally (see Appendix 3); probabilistic analyses aimed at estimating a failure rate do not apply to the evaluation of the correctness of the logic (see Appendix 4) and the analyses of failure propagation used successfully for hardware do not apply to the logic either (see reference [14]). This is why manufacturers, utilities and technical bodies such as IRSN have recognized the need to define a specific approach for the design of the digital instrumentation and control systems of nuclear reactors to be able to demonstrate the correctness of their logic. Independent verification and validation, according to a technically justified strategy, constitute a complementary precaution.

This approach is supplemented by functional diversification that allows to overcome a hypothetical specification or design fault of certain functions by means of other functions achieving the same objectives by using different

physical signals and different processing. Moreover, a hypothetical technological failure of a family of computers is overcome by a means based on different mechanisms and different software and hardware components.

This approach has evolved over time to take into account technological evolutions such as network communications, as well as scientific and technical progress such as “formal” verification methods based on a mathematical approach. It is fully consistent with the international consensus expressed in the IAEA texts (safety guide SSG-39 relating to instrumentation and control) and the IEC texts (standards concerning nuclear power plant instrumentation and control), and similar to those in other industrial sectors where instrumentation and control performs important to safety functions, such as avionics, space and rail.

The relevance of this approach is confirmed by the experience feedback from the digital safety systems developed for over 30 years (see Appendix 2).

However, it would not be desirable to increase the complexity and the integration of instrumentation and control beyond what is done for the Flamanville 3 EPR reactor, as it could then become too difficult for current methods to demonstrate the proper completion of the required actions and the avoidance of generalized spurious actions.

The digital systems are organized into a preexisting generic part (basic hardware and software) and a part specific to the application in question. This enables anticipating the technical evaluation of the generic part, often more complex than the specific part from the point of view of instrumentation and control.

APPENDIX 1 - EXAMPLES OF DETAILED DESIGN REQUIREMENTS

The two examples below illustrate the way in which top-level design principles, such as those relating to avoidance of CCFs (see paragraph 6.4.1), are broken down into detailed requirements.

Prevention of CCFs by propagation or interaction

Communication between computers is essential (for example, four redundant computers have to transmit their results to a fifth one responsible for the vote), but failure of a computer shall not disrupt operation of another one, even if they communicate through a transmission network.

Hardware means such as fiber optic links can be used to prevent CCFs that may result from the propagation of physical disturbances; in addition, the design shall prevent CCFs that may result from logic coupling: a computer shall not be disrupted if one with which it exchanges information no longer communicates or communicates incorrectly. Thus, for systems classified 1E (or F1A):

- operation of the transmission networks shall be deterministic: the necessary resources such as access to the physical transmission medium or to the memory used in the transmitting and receiving equipment, as well as the calculations done in this equipment, shall be predefined and shall not, in particular, depend on the transmitted values or variations in the process;
- the transmission on a network shall be done cyclically, without taking into account the status of the receiver(s) and without even knowing this status;
- reading on a network shall be done on the initiative of the receiving computer, without knowing the state of the transmitter and without a wait resulting from the network status;
- this reading shall be possible at any moment and shall provide correct values, including when a transmission on the same network occurs simultaneously.

For example, a computer A can acquire and transmit cyclically on a network the values of two sensors X and Y. These values are received by a computer B, which uses them in a calculation for which the accuracy needs the values of X and Y to have been acquired very close together.

B have to be able, at any moment, to read the values transmitted by the network, without waiting, in order to preserve the deterministic nature of its own operation. If this reading is done at the same time as the transmission by A, just after the transmission of X and just before that of Y, the most recent values available for B are temporally inconsistent as the one for X is very recent and the one for Y dates from the previous transmission cycle. To guarantee consistent calculations, the interface with the network have to provide B with the most recent consistent set (in this case, the values of X and Y from the previous transmission cycle) and not the most recent individual values.

If these conditions are met, the failure of a transmitting computer may result in incorrect data being transmitted to the network, an absence of transmissions or even monopolization of the network ("jabbering"), but operation

of the receiving computers is not affected: they read the data received, which may be incorrect or absent, and continue their calculation cycle normally. This enables the absence or incorrect nature of the data to be tolerated by means of redundant data from different networks subject to an appropriate selection, such as the choice of the median value for numerical data or the vote for binary data. The calculations can then continue undisrupted using this "consolidated" data.

Therefore, the failure of a computer producing the partial safety action demand in a train cannot spread to the computer performing the vote between the partial demands of all redundant trains, provided that these demands are transmitted by separate networks in accordance with the conditions mentioned above. Similarly, failure of the voter cannot spread to the computers producing the partial safety action demands, which enables these computers to also transmit their requests, undisrupted, to a redundant voter.

Prevention of CCFs by coincidence

The redundant computers could fail at the same time if their operation depended on external phenomena such as transient conditions of the process and if one of these phenomena activated a design fault. For example, standard industrial computers are often configured to transmit the values of sensors only if they change, in order to minimize network load: the transient conditions then induce transmission peaks that could saturate the transmission and calculation capacities of redundant processing units.

The design shall prevent this type of CCF by guaranteeing operation of safety computers independent of their environment. Thus:

- a computer shall read the values given by the sensors (or received from other computers), do its calculations and transmit the orders to the actuators (as well as the data to other computers) cyclically, at its own pace, invariantly and independently of its environment, in all of the situations in which the functions it performs are required;
- the number of pieces of data processed in each cycle (sensors, actuators, communications) and the sequence of calculations shall not vary and shall not depend on the date values;
- operation of a computer shall not depend on any other external conditions other than those specified (for example, inactivation of a computer with a physical key, under control of an appropriate procedure);
- operation of a computer shall not depend on any events or information not specified, such as specific calendar dates or the number of calculation cycles completed since the computer was restarted;
- operation of the computer's infrastructure (reading of inputs and writing of outputs, communications, cycle management, self-monitoring, etc.) shall not depend on the instrumentation and control functions that it performs or their data.

Under these conditions, the sequence of computer operations does not vary and is independent of its environment, which prevents the latter trigerring the CCF of several computers.

APPENDIX 2 – EXPERIENCE FEEDBACK FROM DIGITAL SYSTEMS

As soon as the use of digital technologies was considered for safety systems, operators, manufacturers and technical bodies such as IRSN identified the need for an approach that was different from common industrial practices, targetting the absence of faults. Experience feedback from the digital systems developed using this approach seems to be very positive: for example, the digital protection systems in French power plants have not experienced unsafe failures caused by a software fault.

Analysis of the experience feedback presented in reference [17] concerns around ten million hours of PS computer operation. It shows that the main contribution to latent CCFs (i.e. faults that would have caused CCFs if certain initiating events had occurred) is maintenance and not design: of the 26 latent CCFs identified, most concern calibration, incorrect adjustment or, more generally, maintenance errors. Anyway, maintenance errors are not specific to digital systems which, on the contrary, help to prevent them. A single latent CCF resulted from a software fault, introduced during a modification that involved adding diversified sensors to the sensors already present in the redundant trains of a protection system. The logic added to manage these sensors was incorrect for a scenario of specific failures of the two diversified sensors in the same train: it provided a measurement incorrectly considered to be valid despite the detection of failures affecting both sensors. This illustrates the risk of reducing the quality of the instrumentation and control while believing that it is being improved, as a result of increased complexity.

The Electric Power Research Institute (EPRI) studied experience feedback from safety systems operated in the USA for over 20 years and has drawn a similar conclusion [18]: of the 49 incidents that affected these systems, 6 could have led to a CCF and only one was due to a software fault. There are two important lessons to learn from this incident:

- the fault is due to the excessive and pointless complexity of the software, which managed self-monitoring sequences depending on the position of a switch;
- the design of this software does not comply with the principles described in this document concerning cyclical performance and operation independent of the environment, which confirms the soundness of these principles.

The two aforementioned cases of latent CCFs caused by software, which are marginal compared to other failure scenarios, come partly from excessive requirements, intended to improve safety, but which made the system more complex and in the end produced the opposite effect.
The usefulness of back-up systems shall therefore be evaluated taking into account the increased complexity of instrumentation and control, in particular of its maintenance that so far seems in practice to be the main contributor to CCFs of safety instrumentation and control.

APPENDIX 3 - KNIGHT AND LEVESON EXPERIMENT

Software diversification involves executing in parallel several programs that implement the same function but that were developed independently, based on the assumption that the faults they may have do not concern the same scenarios and can therefore be masked by an appropriate vote.

Knight and Leveson conducted the experiment described in reference [16] to assess the relevance of this fault independence hypothesis: 27 versions of a software program meeting precise functional requirements were developed by 27 different people.
Each version was tested by comparing it with a reference version for one million scenarios. As the authors reported, this method may have concealed faults common to all of the versions and the reference, and was therefore able to influence the results in favor of the diversified approach.
Other conditions of the experiment were favorable to this approach. For example, the mathematical nature of the requirements (favoring the absence of ambiguity) and the absence of implementation constraints (concerning the memory, the execution time, the output format, etc.) eliminated potential sources of faults common to several versions.

The 27 versions were overall good. The tests revealed 1.6 failures per version on average, four and seven failures in the two worst ones, and no failure in the six best ones.
Despite the favorable conditions, the experiment showed a number of multiple failures (affecting several versions for the same input) incompatible with the hypothesis of statistical independence that motivated the approach. This hypothesis therefore had to be rejected with 99% confidence.
No correlation likely to skew the experiment was found between the failures of the versions and the profiles of their authors (training, experience, location, etc.).
The diversification of a software program cannot therefore, without further evidence, be considered as a measure to mask the presence of hypothetical faults.

APPENDIX 4 – PROBABILISTIC APPROACHES

The ability of a software program to accomplish its function is the result of its design and does not change during its service life: logic cannot therefore fail, strictly speaking, as failure is defined as loss of this ability (see paragraph "Definitions"). The expression 'failure probability' is therefore not suited to logic.
However, some bodies put specific requirements concerning the quantification of the probability of failure due to software in instrumentation and control systems. Such specific requirements have not been included in the IAEA instrumentation and control guide [7] or in the standards of IEC committee 45A ([8] to [13]).
As statistical testing is often considered to get such figures, it is presented in brief below.

Statistical testing and operational profile

Statistical testing aims to estimate, not the ill-defined failure probability of the logic, but the probability that, in a given environment, the inputs applied to the logic activate a fault: the logic is tested by sequences of inputs drawn "randomly" from the "operational profile", defined as all of the independent sequences of inputs, weighted by their frequency of occurrence in the environment in question.

One key point is that this probability is equal to the total weight of the sequences for which the logic is faulty. Let us assume that a given logic is wrong for a particular input sequence; if this sequence appears half of the time in a certain environment, the failure probability equals one half; if it appears one time in a billion, the failure probability is one billionth, for the same logic and the same fault.
The confidence in the estimation of this probability therefore depends directly on the confidence in the identification and weighting of the independent sequences of inputs produced by the environment in question.

As indicated in this document, there are many more possible input sequences for an instrumentation and control system than there are atoms in the universe. It is extremely difficult to know which are actually possible in a given environment and to weight them according to their frequency in this environment, with the precision required to obtain a credible estimation of the probability sought.

Practical impossibility of knowing the operational profile

It has been proposed to identify the operational profile by observing the inputs of an identical system that is already in operation, but this is not possible for a new reactor. In addition, the outputs generally depend on the current inputs but also on memories that may have been set to different values in the past, at unknown times; for example, the exceeding of a threshold by a signal can produce different values of a given output depending on whether or not a certain event has been detected before, even if the occurrence of this event has not modified this output in the meantime and, as a result, nothing indicates this dependency to an observer. The continuous input history cannot therefore be arbitrarily divided into independent sequences to create a correct operational profile.
Lastly, the number of possible sequences is such that very few of them will be observed in practice: thus, an operational profile based on the observation of all of the reactors in the world since their commissioning would not even contain one scenario for each type of accident, which raises concerns when the object to be tested is a protection system whose main mission is to detect accidents.

It has also been proposed to identify the operational profile from the physical signals used in the accident studies. But as a physical signal such as the CHFR is calculated in the protection system using hundreds of inputs and parameters, a huge number of input and parameter combinations lead to the same CHFR value. The CHFR value at a given time in an accident study does not therefore enable to determine and weight all of the physically possible input combinations corresponding to this value. And yet, as the calculation may be incorrect for one combination and not for the others, they cannot be considered as equivalent and they should therefore be correctly weighted to make the estimated probability objective; as mentioned above, if the faulty combination appears one time in a billion (respectively half of the time) in the actual environment, the failure probability is one billionth (respectively one half). Knowing the actual weight of each input sequence is still mandatory.

Conclusion on statistical tests

No means have yet been proposed to know precisely the operational profile of an actual environment (completeness of scenarios and accuracy of weighting) and therefore to carry out a relevant statistical testing. The "probability that, in a given environment, the inputs applied to a system activate a logic fault" makes sense, but none of the proposed means provides a credible estimation.

The safety software reference standard [10] summarizes this situation: "*The validity of the calculated (probability) depends upon the similarity of the profile of the test inputs to the profile of the actual inputs experienced by the system in operation. If (…) used on an unrealistic operational profile (…) a (probability) will be estimated that may be very different to the actual system availability (…). This is a fundamental weakness of the statistical testing approach as it is generally very difficult to accurately determine the operational profile that a system will experience in use, and this is particularly true for systems with large numbers of inputs*".

Probabilistic safety assessments

For the probabilistic safety studies, consensus values of failure probability, justified qualitatively, can be used for the instrumentation and control systems. For example, the nuclear instrumentation and control experts from IEC committee 45A estimate [8] that "*For an individual system, (…) a figure of the order of 10^{-4} failure/demand may be an appropriate overall limit to place on the reliability that may be claimed, when all of the potential sources of failure due to the specification, design, manufacture, installation, operating environment, and maintenance practices, are taken into account*". The combination of independent systems helps to improve this reliability (e.g. the protection system of the EPR actually consists of two independent subsystems, performing diversified automatic scram functions, to achieve a probability in the region of 10^{-5} failures per demand).

The same approach is used in other industrial sectors such as aviation technology, space, rail, automation, automotive, etc. [19]

APPENDIX 5 - SPECIFIC TECHNIQUES (INTERRUPTS)

Standards such as [7], [10], [11] and [12] describe specific implementation techniques, which may be suitable in certain cases, but not in others, depending on details concerning the functional requirements, the architecture of the system and hardware operation. This document does not seek to analyze these details and as a result does not adopt a position on the corresponding implementation techniques.
This Appendix illustrates this subject using the key example of "interrupts". An interrupt is a hardware signal transmitted to a processor, which forces it to suspend the current processing ("main" processing in the rest of this Appendix) to execute another one before resuming the main prosessing.

Interrupts can be transmitted to the processor, for example when a piece of information arrives from a network, so that the processor can save it and process it. Interrupts can also be transmitted to the processor at predefined intervals, for example so that it performs self-monitoring during some of its time.
In general, the main processing depends on the data handled during the interrupt: this is the case, for example, when this processing involves calculating a function which uses measurements from sensors and networks, and if the variables containing these measurements are updated by way of interrupts. The variables used by the main processing can then be modified at any time by the interrupts.
Incorrect results can be produced if precautions are not taken. Let us consider, for example, a variable of the main processing represented by two characters, equal to "09" at a given moment. The two characters "1" and "0" forming the new value "10" are then transmitted one after the other to the processor and each one generates an interrupt. If the processor uses this variable before the first interrupt, it performs its calculations with the value "09"; if it uses it after the second interrupt, it calculates using the value "10", which is also correct and corresponds to a sampling time slightly later in the process. But if it uses it between the two interrupts, it calculates using the value "19", which does not correspond to any actual state of the process and produces an incorrect result.
This simplified yet representative example illustrates the need to take precautions when two different parts of the processing are using the same data or, more generally, the same resource. In an uncontrolled design, it is very difficult to avoid such synchronization errors; this has given interrupts a bad reputation and some texts even suggest banning their use.

However, in a safety system, several computers generally have to exchange information, and they usually operate asynchronously to avoid the propagation of failures (each one transmits to the other without knowing its state). For a given computer, the reception of information and the performance of calculations that use this information are therefore mutually asynchronous; thus two asynchronous processing parts are needed to deal with these two aspects.
As indicated above, using an interrupt satisfies this need by enabling the same processor to perform the main processing and to save the data received when it arrives, so that it can be used in the next calculation cycle.

However, if these processing parts are not properly synchronized, inconsistencies may arise, as the example above shows.

Another way of performing the two asynchronous treatments involves using, in addition to the processor responsible for the main processing, an auxiliary processor to receive the information and store it in a “double access” memory; the main processor also accesses this memory, for example at the beginning of each calculation cycle, to read the stored information and use it in its calculation. This “shared memory” technique avoids interrupts, as each processor is responsible for one part of processing only; it is therefore sometimes recommended in documents that choose to impose implementation choices. However, used carelessly, the “shared memory” technique suffers from exactly the same weakness as the interrupt-based technique: in the example above, if the main processor reads the two characters while the auxiliary processor is modifying them, inconsistency may arise.

We do not therefore wish to either recommend or prohibit either of these techniques, but rather recommend properly solving the underlying problem associated with parallel execution of two interdependent processes. The principle of such a resolution, known since the work of Edsger Dijkstra (Solution of a Problem in Concurrent Programming Control, Communications of the ACM, 1965), relies on the use of rigorous protocols to attribute to one process at a time the right to use a common resource. As a mathematical principle, it applies to one technique as well as the other: a designer able to correctly solve the problem with a shared memory is also able to solve it with an interrupt. One or other of the solutions can therefore lead to a simpler design that is easier to verify, depending on details such as the quantity of data to be transferred, its arrival frequency in relation to the main processing frequency, the processor load, etc.

IRSN
INSTITUT
DE RADIOPROTECTION
ET DE SÛRETÉ NUCLÉAIRE

Headquarters
31, avenue de la Division Leclerc
92260 Fontenay-aux-Roses
RCS Nanterre B 440 546 018

Phone
+33 (0)1 58 35 88 88

Mail
B.P. 17 - 92262 Fontenay-aux-Rose Cedex - France

Website
www.irsn.fr
@IRSNFrance, @suretenucleaire