Transactions of the Korean Nuclear Society Spring Meeting
Kyungju, Korea, May 29-30, 2008

Countermeasures of SMART Digital Plant Protection Systems (DPPS) against CCF

Jong Yong Keum, Gwi Sook Jang, Yong Suk Suh, Heui Youn Park
I&C and HF Div., KAERI, 150-1 Dukjin-dong, Yuseong-gu, Daejon, Korea, 305-353, jykeum@kaeri.re.kr

1. Introduction

Common Cause Failure (CCF) is the concurrent failure of two or more functional units (structures, systems or components) due to a single specific event or cause. This CCF can not credit an independence of redundant channels of safety equipment or multiple critical systems in nuclear power plants. This paper presents countermeasures of System-integrated Modular Advanced ReacTor (SMART) DPPS against CCF caused by digital faults for the purpose of meeting an independence of redundant channels of SMART DPPS.

2. Common Cause Failure Mechanism

Fig. 1 shows a CCF mechanism. Digital failure is a systematic failure resulting from the activation of a digital fault. Digital faults and triggering conditions are indispensable conditions which lead to digital CCFs. The effect of a digital fault is latent but contributes to a failure mechanism. If a triggering condition is satisfied, a digital fault results in a digital failure.

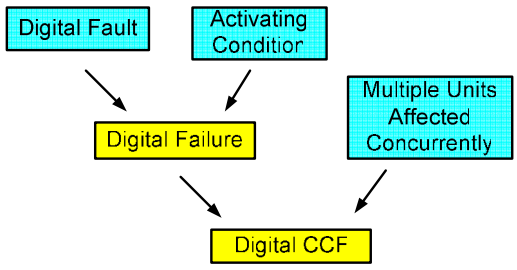

Fig. 1. CCF mechanism

3. Digital fault classes and their failures

In this paper, it is assumed that the defenses against digital CCFs focuses on digital faults. These digital faults can be categorized into four classes - specification faults, hardware faults, software faults, and hardware-related software faults. Fig. 2 shows the digital system failure categories resulting from these digital faults

3.1 specification faults

In high quality software, digital failures tend to be predominantly caused by specification faults. As a main cause of them, Thuy [1] indicates a lack of understanding - specifiers do not fully understand or know the context of an Instrumentation & Control (I&C) system. This specification fault produces wrong specification for the requirements, architecture, and design of a system from real world requirements. The focus should be on unsafe system failures, not software errors. Error-free software can easily introduce unsafe behaviors (not necessarily failures) if it is built to a flawed requirements specification [2]. Therefore, the validation of functional specification prior to a software implementation and testing is very important in reducing a CCF.

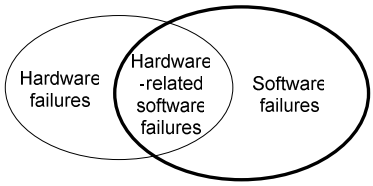

Fig. 2. System failure categories

3.2 hardware faults

Keene and Lane [3] investigate the failures of the same type of circuit cards. The interesting observation was that these failures did not fit the random failure models. These failures were deterministic and resulted from a defect in a design. Thus, a design defect is also an important cause of hardware failures. If this hardware design defect occurs at the same time, it may disable redundant channels of digital I&C safety systems. Defensive measure against hardware design defect is design diversity because different designs will have different failure modes and will not be susceptible to the same common influences. And as an alternative to design diversity, equipment diversity (e.g., Intel 80x86 architecture versus Motorola 68000) can be selected.

3.3 software faults

Software faults (or hardware-unrelated faults) are faults in those software modules that are unrelated to the hardware components. These faults are 'pure' software faults.

Software design errors and coding errors are main factors of CCFs due to a software. The most effective diversity against these CCFs is a human diversity and next is a software diversity. Although diverse software versions are developed by using different specifications, designs, programming teams, programming languages, etc, many researchers have revealed that those independently developed software versions do not necessarily fail independently [4].

3.4 Hardware-related software faults

Hardware-related software failures are mostly caused by the degradation of hardware and the design defects of software. In this case, hardware components are partially failed, but the system can still perform normal operations (in a degradation manner) if workload is not very high. However, if workload is high for the system

and the hardware degradation is undetected by the software, then the software may try to perform operations on the failed hardware components and the system fails. This kind of software failure is considered as a hardware-related software failure. A good software design should avoid hardware-related software failures.

4. Countermeasures of SMART DPPS against CCFs

As countermeasures of CCFs against wrong functional specifications which may be caused by MMIS (Man-Machine Interface System) Design Team, independent reviews of functional specifications by other design groups (Core Design Group, Fluid System Design Group, and BOP Design Group) which provide the functional requirements are performed during SMART DPPS development [5].

Currently, each channel of the SMART DPPS adopts a Digital Signal Processor (DSP)-based hardware platform. Software modules which perform safety functions were developed with assembly language in favor of a verification and validation of software codes. And each channel will be designed to be diverse dual processors. One of diverse dual processors is TMS320Cx processor of Texas Instruments Co., the other is SHARC processor of Analog Devices, Co.

For a software diversity, two different operating systems can be selected to prevent a CCF which is latent in an operating system in digital I&C system adopting diverse dual processors inside each channel. But such a selection can increase a system complexity due to interactions between an operating system and hardware, and between an operating system and safety system software modules.

Because generally, companies of commercial operating systems do not open their sources, a verification & validation of operating system in safety aspects is very difficult. So, in SMART DPPS, very simple deterministic scheduling software modules were developed and used instead of an operating system. This shows specific design features (no use of operating system) can be used instead of software diversity (use of different operating systems in this case) to assure an independence. Additionally, by no use of operating system, we can have advantages to avoid a system complexity and to facilitate a CCF analysis.

The CCFs due to hardware-related software faults may be protected by good software design features. That is, these design features in Table I may protect the CCFs caused by interactions between hardware and software.

5. Conclusions

In brief, we looked into the classes of digital faults and countermeasures against them. Following conclusions are deduced:

(1) Based on an appropriate engineering judgement, determining what types of design features and diversities against CCFs are effective in reducing the likelihood of CCFs;

(2) the use of diversity does not assure an independence of redundant channels. In order to assure the independence, specific design features must be credited in addition to diversity;

(3) diverse dual processors and two other operating systems can increase system complexity.

Table I : Countermeasures of SMART DPPS against CCF

	Countermeasures
Specification fault	Independent Reviews with other Design Groups
Hardware fault	equipment diversity - TMS320Cx processor-based DSP - SHARC processor-based DSP (variable)
Software fault	Extensive V&V Different program languages - assembly language - C/C++ language (variable)
Hardware-related software fault	Defense against this fault may not need diversity. Only software design features listed below may be required. - Minimal use of interrupts - Avoidance/removal of deadlocks

REFERENCES

[1] N. Thuy, “Defense against Digital CCF”, IAEA Technical Meeting on Common-Cause Failures in Digital Instrumentation and Control Systems of Nuclear Power Plants, Bethesda, Maryland, USA 19-21 June 2007

[2] Ray Totro, “Integrated strategy for managing vulnerability to common cause failures”, IAEA Technical Meeting on Common-Cause Failures in Digital Instrumentation and Control Systems of Nuclear Power Plants, Bethesda, Maryland, USA 19-21 June 2007

[3] S. Keene and C. Lane, “Combined Hardware and Software Aspects of Reliability”, Quality and Reliability Engineering International, Vol.9, 1992, pp 419-426

[4] Yu Hayakawa, Telba Irony, and Min Xie, “System and Bayesian Reliability”, World Scientific Publishing Co. Pte. Ltd., 2001

[5] I. S. Koo, “Design of Man-machine Interface Systems for SMART considering CCF”, IAEA Technical Meeting on Common-Cause Failures in Digital Instrumentation and Control Systems of Nuclear Power Plants, Bethesda, Maryland, USA 19-21 June 2007