Progress in Nuclear Energy 70 (2014) 128–133

Contents lists available at ScienceDirect

Progress in Nuclear Energy

journal homepage: www.elsevier.com/locate/pnucene

Review

Quality assurance for a nuclear power plant simulator by applying standards for safety-critical software

Ye Cheng [a, b, *], Ni Chao [b], Zheng Tian [b], Zhang Zhicheng [b], Zhang Ronghua [b]

[a] *School of Nuclear Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China*
[b] *Shanghai Nuclear Engineering Research and Design Institute, Shanghai 200233, China*

ARTICLE INFO

Article history:
Received 9 January 2013
Received in revised form
2 September 2013
Accepted 5 September 2013

Keywords:
Nuclear power plant simulator
Quality assurance
Standards for safety-critical software

ABSTRACT

Nuclear power plant simulators are playing a more important role in nuclear power plant lifecycle analysis, and the quality of the simulators should be verified to ensure the safety of nuclear power plants. Currently, there is no systematic quality assurance method for nuclear power plant simulators. In this paper, a systematic quality assurance method for nuclear power plant simulators is proposed basing on experiences with safety-critical software. Key aspects of the method are discussed. In addition, application of this method to a real project is also described as a practical reference.

1. Introduction

The application of simulation technologies in Nuclear Power Plants (NPPs) has a very long and successful history. NPP simulators, including traditional training simulators, NPP analyzers, and various simulators with specific purposes, are the major applications of simulation technologies in NPPs. In addition, NPP simulators are playing a more important role in the entire NPP lifecycle, beyond traditional training and education (Bartsoen, 1997; Izquierdo et al., 1993; Juslin et al., 1997; Kim et al., 2007; Kim and Rizwan, 2007; Pochard et al., 2002). In Fig. 1, typical applications of NPP simulators in the NPP lifecycle are listed. Principle simulator can cover all the NPP lifecycle; analyzer can support design, construction and research; engineering simulator can verify design and construction; full scope simulator can be used for operator training. When a NPP simulator is applied in research, design or operation of NPP, its impact on the NPP's safety should be seriously considered. Only if the quality of the NPP simulator is assured with a high level of confidence can the quality of research, design, or operation involving the NPP simulator be assured. According to NQA1, which is the nuclear quality assurance requirements made by, ASME, computer program must be preverified for design control (American Society of Mechanical Engineers, 2008). When an NPP simulator is used in design activities, a systematic quality assurance method should be used to support its preverification.

NPP simulators do not operate as components in any nuclear facility and would not lead to immediate damage to a NPP. Any result generated by simulator is used in NPP research, design or operation activities. i.e., the correctness of simulation results is the most important. The correctness of an NPP simulator is decided by 3 factors: the appropriateness of the mathematical model for the physical NPP problem to be simulated, correct construction of the NPP simulator to produce a correct solution for the encoded mathematical model, and correct use without introducing human error. Application experiences with NPP simulators show that the widely used mainstream mathematical models are adequate for a successful NPP simulation, while the construction and use of NPP simulators are case specific and are highly dependent on the developer and the user of the NPP simulator. Currently, an NPP simulator's correctness is primarily assured by verification and validation activities. ANSI/ANS-3.5 (2009) is the only formal standard that can be referenced for guidance. In this standard, verification testing, validation testing, and performance testing are defined, and criteria are provided. ANSI/ANS-3.5 aims at meeting the requirements of code 10 CFR 55 (Code of Federal Regulations, 1998) of the Nuclear Regulatory Commission (NRC), rather than being used for general-purpose NPP simulators.

Generally, an NPP simulator is composed of high-performance computers and a human-machine interface (hardware panels or soft panels), as shown in Fig. 2. A complicated simulator software system runs on the computers. The software system implements

* Corresponding author. School of Nuclear Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China. Tel.: +86 18601728652.
E-mail address: yecheng@snerdi.com.cn (Y. Cheng).

0149-1970/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.pnucene.2013.09.005

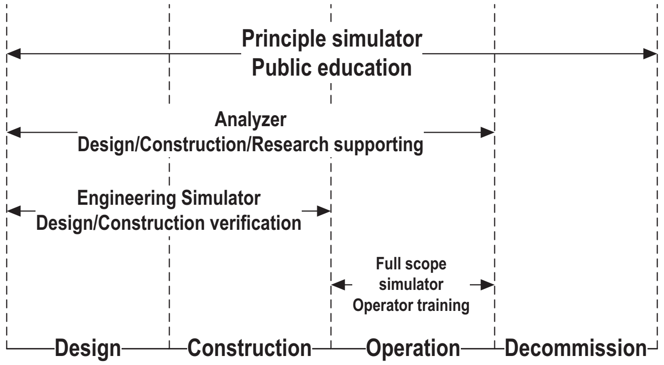

Fig. 1. Application of a simulator in the NPP lifecycle.

multiphysics process simulations (heating, flow, neutron kinetics, control, human-machine interaction), data communication, task scheduling, and the supporting functions necessary for a complete simulation. The hardware is relatively simple and mature, while the software is the source of quality issues. Errors could be introduced in every stage of the entire lifecycle of NPP simulator software. Some of these errors would cause the NPP simulator to crash and produce no simulation result, while others would lead to fake simulation results. The latter should be avoided or eliminated to ensure the correctness of the NPP simulator. Experiences in the software industry show that a single technique, such as verification and validation by testing, as described in ANSI/ANS-3.5, is not enough for NPP simulator quality assurance; a systematic method should be employed.

In this paper, applying standards for safety-critical software to NPP simulators as a systematic quality assurance method is proposed and discussed.

2. Standards for safety-critical software

2.1. Software technology

In the software industry, the campaign for high-quality, complicated, large-scale software has lasted for several decades, and there seems to be no end to this trend in sight. Unlike other industries, software is mainly produced manually by humans (some CASE-Computer-Aided Software Engineering tools can automate a small portion of the software development work, but CASE is only available in special domains or cases) and is always error-prone. Structure programming, software engineering, object-oriented programming, the software capacity model of maturity, etc., were innovative, and these techniques or methods did move the software industry forward. However, the perfect solution is still missing (Brooks, 1987). There is no easy way out for producing software.

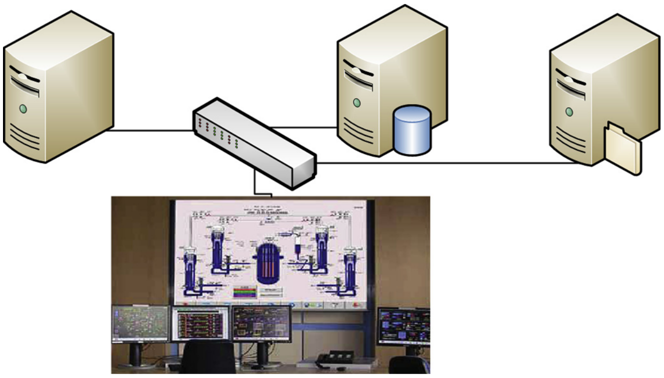

Fig. 2. Hardware composition example for an NPP simulator.

2.2. Safety-critical software technology

In safety-critical fields, such as NPPs, aerospace applications, and high-speed railways, digital technology is employed extensively, and software is becoming more and more important. In many cases, some key functions can only be implemented by software. When software is used in these safety-critical fields, the impact of the software on safety is treated as a key issue and taken seriously, and a considerable amount of effort is devoted to developing the software. Remarkable developments and achievements in the above safety-critical fields show that safety-critical software (i.e., software used in safety-critical applications) has no negative impact on safety, and the safety targets are achieved.

In safety-critical fields, industry, academia, and the regulation authorities work together and come to a consensus, stating requirements/guides clearly in safety-critical software standards. Examples are DO-178B for civil aviation (RTCA/DO-178B, 1992), EN 50128 for railway applications (EN 50128, 2001), and IEC 60880 for NPPs (IEC 60880, 2006). Although there are obvious differences among these safety-critical software standards, they share a common methodology, i.e., by defining a framework for the software lifecycle with necessary processes and associated requirements, safety-critical software standards provide a systematic method to avoid and eliminate errors throughout the lifecycle of safety-critical software. In these standards, the software lifecycle is generally composed of engineering processes, such as the software requirement process, software coding process, and supporting processes, such as the software planning process and the software configuration process. In addition, requirements for each process are also given to guide practices.

2.3. NPP safety-critical software standards

For safety-critical software used in NPPs, the regulation authority (In China, the Chinese National Nuclear Safety Administration, NNSA, is the regulation authority) provides regulation guides according to the law and expresses their viewpoint on safety-critical software. In regulation guides, certain industry standards are endorsed as official standards. Fig. 3 shows the NPP software standard system in China. Software providers and users develop and use software according to the standards endorsed in regulation guides and provide evidence that all requirements have been met, and then acceptance by the regulation authorities can be obtained. From a technical point of view, such practices can assure safety with a high level of confidence.

There are several NPP standard systems for safety-critical software in NPPs, including the IEEE series (IEEE 7-4.3.2, IEEE 1012, IEEE 1074, etc.) and the IEC series (IEC 60880, IEC 62138, etc.). In this paper, IEC 60880 is taken as an example to discuss how to apply safety-critical software standards to NPP simulators.

2.4. IEC 60880 briefing

IEC 60880 provides requirements for the software used with the computer-based instrumentation and control (I&C) systems of NPPs that perform functions from safety category A (as defined in IEC 61226) that must be met to produce highly reliable software. It addresses each stage of software generation and documentation, including requirement specifications, design, verification, validation and operation. The software lifecycle processes defined include