SMART-P MMIS Software Development by Considering the Software License for Nuclear Power Plants and the Development Cost

Yong Suk Suh,[a] Jae Hong Park,[a] Heui Youn Park,[a] Ki Sung Son,[b] Ki Hyun Lee,[b] Hyeon Soo Kim [c]
a Div. of I&C and HFE, KAERI, 150 Dukjin-dong, Yuseong-gu, Daejon, Korea, 305-353, yssuh@kaeri.re.kr
b Control Tech. Research Inst., SEC Ltd.,974-1 Goyeon-ri Woongchon-myon, Ulju-gun, Ulsan, Korea, 689-871
c Dept. of Computer Science and Eng., Chungnam Nat'l Univ., 220 Gung-dong, Yuseong-gu, Daejon, Korea, 305-764

1. Introduction

The acceptance criteria of software for safety system functions in NPPs (Nuclear Power Plants) are as follows: 1) acceptable plans should be prepared to control the software development activities, 2) the plans should be followed in an acceptable software life cycle, and 3) the process should produce acceptable design outputs [1]. The KINS (Korea Institute of Nuclear Safety) recommended that the software life cycle should be established based on the IEEE Std 1074 with a supplementary requirement of a software safety analysis. The KINS emphasized that the software should be developed to show its high qualities [1]. This paper identifies the major requirements to achieve the software license from the KINS and presents the major facts reflected in the SMART-P (System-integrated Modular Advanced ReacTor-Pilot) MMIS (Man-Machine Interface Systems) which is being developed by KAERI and targeted to start operation in 2010. This paper also addresses major concerns on the development of a safety critical software and the facts reflected in the SMART-P MMIS.

2. Major Facts in the Software Development

After reviewing the software development requirements [1], this paper categorizes them as follows: 1) software classification, 2) software development life cycle, 3) software quality assurance, 4) software configuration management, 5) software hazard analysis, 6) software verification and validation, and 7) use of a pre-developed software. The KINS recommended that the IEEE standard should be referenced to perform the software development activities.

2.1 Software Classification

There are three grades of software classification in the SMART-P MMIS: a safety-critical (SC) software, safety-related (SR), and a non-safety (NS) software. The classification is assigned to each system in the SMART-P MMIS based on its criticality as shown in Table 1.

2.2 Software Development Life Cycle (SDLC)

The SDLC is comprised of eight phases: an initiation, plan, requirement, design, implementation, integration and validation, installation, and an operation and maintenance [2]. The SDLC is sequentially performed in accordance with the waterfall model. The SDLC complies with the IEEE Std 1074 as shown in [2].

2.3 Software Quality Assurance

The software quality assurance plan and procedure complying with the criteria in KEPIC QAP-1 and QAP-2.7 and IEEE Std 730 are performed throughout the SDLC. Software development organization consists of a project management team, quality assurance team, development team, verification and validation team, configuration management team, and a safety analysis team. The responsibility of each team is clearly defined in its planning documents. Since a document-based software development principle is adopted, all of the software activities are recorded, reviewed, approved, documented, and maintained throughout the SDLC. Proper guides are made to keep a consistency among the software products. The guides refer to the IEEE software standards. The IEEE software standards referenced in the SMART-P MMIS software development are presented in [3]. The software abnormalities are controlled by the quality assurance team until they are fixed.

2.4 Software Configuration Management (SCM)

All the baselined software products are maintained and changed by the configuration control procedure throughout the SDLC. They are all identified with a unique name and revision number. The configuration control board analyzes an impact of the software change and makes a decision on the performance of the change.

2.5 Software Hazard Analysis (SHA)

The SHA identifies abnormal conditions and events (i.e., hazard) caused by a software throughout the SDLC. The SHA consists of the preliminary hazard analysis, fault tree analysis, and the failure modes and effects analysis. The SHA is performed from the viewpoints of a human error, development process error, and a product error.

2.6 Software Verification and Validation (SV&V)

The evidence of the SV&V activities is an important factor to achieve the software license from the KINS. The software faults should be found through the SV&V. The

SV&V is manually performed throughout the SDLC. The V&V activities comprise the software analysis, software review, and the software test. For the software analysis, the traceability analysis, interface, data flow, control flow, and the timing analysis are manually performed. The requirement traceability matrix is maintained throughout the SDLC. For the software review, the management review, technical review, walkthrough, and inspection are performed with a team-based review. For the software test, the software unit test, module integration test, system test, and the MMIS integration test are sequentially performed. All the software tests are procedurally performed: a plan phase, analysis, design, execution, evaluation, and a summary report phase.

2.7 Use of Pre-developed Software (PDS)

The PDS includes a commercial off-the-shelf software, proprietary software, and a previously developed software. The PDS is used after evaluating and assessing that the PDS can perform its required functions.

3. Major Concerns for the Safety-critical Software Development

This paper addresses the major concerns of the KINS for the safety-critical software development as follows: 1) the use of a formal method, 2) software reliability testing, and 3) the software diversity. The formal method such as the use of a mathematical representation or logical proof technique is not adopted due to the following reasons: too many constraints and assumptions are required, it is difficult to understand if a complicated mathematical notation is used, and it will take a long period for the specification [4]. It is not a cost-benefit method. Instead, the technique of a structured analysis and structured design is adopted and conducted using a computer-aided software engineering tool for the analysis of the software requirements and the construction of software architecture.

No software reliability testing is performed because it is not a cost-benefit method. If the loop body has 5 paths and the loop executes 20 times in a module, 5^{20} different paths exist. It may not be realistic to test all the paths. Software failures that are not the consequence of hardware failures are caused by design errors and, therefore, do not follow the random failure behavior used for the hardware reliability [5]. No software diversity technique such as a N-version programming in a system is used. There have been no reports that the technique remarkably increased the software reliability. For the consideration of adopting several different organizations for developing the same software requirement, it will cost much more when compare to the achievements of the software reliability. It is not a cost-benefit method.

4. Conclusion

The software development methodology in this paper was submitted to the KINS which reviews it in terms of safety and will approve its usage for the SMART-P by early 2005. Currently, the SMART-P MMIS is being developed in the “SMART MMIS Joint Research and Development Center” established by KAERI and SEC in 2004. It is true that the method for the software quality metrics and measurement was not shown in this paper. This will be studied further.

REFERENCES

[1] KINS/RR-106, “Development of Safety Requirements and Guides for Digital-Based I&C Systems Important to Safety in Nuclear Power Plants”, KINS, Korea, March 2002.

[2] Yong Suk Suh, et al., “A Development of SDLC for MMIS of SMART Research Reactor”, Proceedings of the Korean Nuclear Fall Meeting, KNS, Korea, October 2004.

[3] Yong Suk Suh, et al., “Software Development Adhering to Software Standards for MMIS of SMART Research Reactor”, Proceedings of 2004 KEPIC-Week, KEA, Korea, August 2004.

[4] KAERI/CM-103/94, “A Study on the Software Development Methodology of a Highly Reliable Software”, KAERI/KAIST, Korea, 1994.

[5] RG-1.152, “Criteria for Digital Computers in Safety Systems of Nuclear Power Plants”, rev 1, U.S. NRC, U.S.A., January 1996.

Table 1 Software Classification of the SMART-P MMIS

Software classification	Safety-critical (SC)	Safety-related (SR)	Non-safety (NS)
Criticality	If this software is performed incorrectly, inadvertently, or out of sequence, SMART-P could be in a hazard state.	This software should have a proper quality to support the SC software.	This software should have a proper quality enough to operate for the non-safety functions in nuclear power plants.
Quality assurance (QA) criteria	KEPIC QAP-1 and QAP-2	Corresponding to KEPIC QAP-1 and QAP-2.7	Complying with industrial standards
Common cause failure (CCF) analysis requirement	Yes	No	No
Independent V&V (IV&V) requirement	Yes	No	No
Electrical classification	1E	Non-1E	Non-1E
Systems in the SMART-P MMIS	PPS, SCOPS, Safety control system, Safety instrumentation, PAM (category A) indication, Safety local controller	DPS, AIS, CFMS, Large display panel (fixed part), Non-safety neutron instrumentation	IPS, NIMS, Large display panel (variable part), Non-safety instrumentation, Non-safety control system, Non-safety local controller