NuDE: Development Environment for Safety-Critical Software of Nuclear Power Plant

Jong-Hoon Lee*, Junbeom Yoo
Division of Computer Science and Engineering, Konkuk Univ., 1 Hwayang-dong, Gwangjin-gu, Seoul, 143-701, Republic of Korea
**Corresponding author: kirdess@konkuk.ac.kr*

1. Introduction

Safety-critical systems are systems where it is essential that system operation is always safe [1]. Therefore, rigorous quality demonstration is important when developing software in safety-critical system. Generally, safety-critical software demands extremely high-confidence verification and validation (V&V) techniques. Additionally, software V&V should be performed in parallel with software development. In IEEE standard 1012 for Software V&V [2], V&V tasks are defined for each development phase.

Software in Nuclear Power Plant (NPP) such as a Reactor Protection System (RPS) is also safety-critical software. RPS makes decisions for emergent reactor shutdown. Therefore, RPS software should be verified strictly and throughout entire development life-cycle. However, it is hard to apply these process and techniques, because the techniques are difficult to understand, the tools often work only in isolation, and the output is difficult to extract meaningful information. In order to overcome these difficulties, we developed a formal methods based process [3]. We also developed tools for supporting formal methods based techniques.

In this paper, we introduce NPP's safety-critical software Development Environment (*NuDE)*, through the Integrated Environment (IE) approach.

2. Formal Methods

When developing and verifying safety-critical software, formal methods based techniques are strongly recommended for increasing safety quality. Formal methods are a particular kind of mathematically-based techniques for the specification, development, and verification of software.

2.1 Formal Specification

A formal specification is a specification expressed in a formal language. Formal specification has several advantages. First, formalization process detects many problems in informal specifications. The requirements of the system are clarified, and ambiguities are revealed. Second, a formal specification can be used to verify that the requirements for the software being developed have been completely specified. Verifying correctness of an implementation by formal verification techniques can be applied in V&V phase in software development.

2.2 Formal Verification

During and after the implementation phase of software development, the results should be checked to ensure that it satisfies its specification. Formal verification is strong static verification technique to check results of implementation. There are two approaches to formal verification, model checking and deductive verification. Model checking techniques use given finite model of the software and formal specification. A model checker performs an exhaustive search of all states the software model. Deductive verification techniques use axioms and proof rules to prove the correctness of software. Generally, deductive verification is more powerful than model checking, but is more difficult to automate. In our former research [3], we determined to use tools already proven effective, model checking technique and model checker.

3. IE Approach

Integrated Environment (IE) approach is developing a software application that provides comprehensive facilities to users for software development. IE approach supports each phase in development phases, requirement, design, and implementation. Software V&V is also supported. The major role of the IE approach is to achieve better integration between each development phase.

4. *NuDE*

For the development of the *NuDE*, we applied IE approach through porting to Eclipse plug-in from existing independent tools. Eclipse Integrated Development Environment (IDE) project is a typical integrated environment [4]. The Eclipse IDE allows the developer to extend the IDE functionality via plug-ins.

Fig. 1 shows development process and tools, which we already developed in former research [3]. We choose three major tools: *NuSRS*, *NuSCRtoFBD*, and *FBDtoVerilog* [5, 6, 7]. These tools support formal specification and verification techniques.

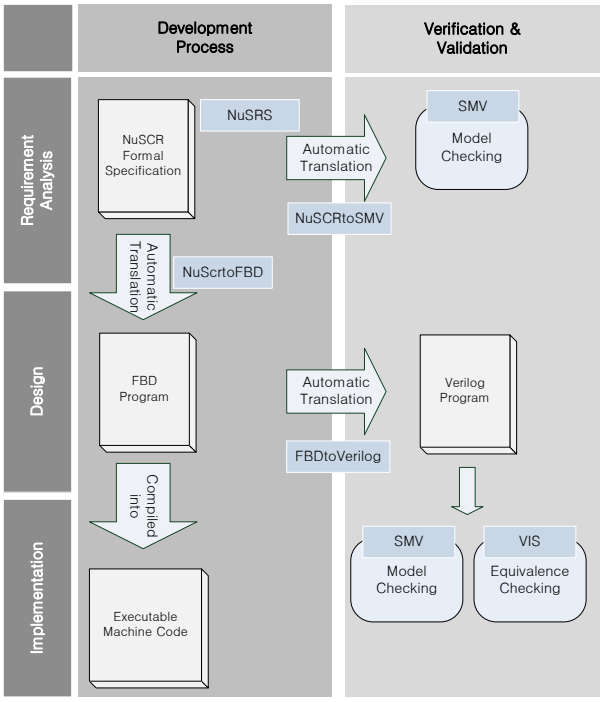

Fig. 1. Development process for NPP's software

4.1 NuSRS

NuSRS is an editor for requirement specification based on NuSCR formal specification language which we developed [8]. During requirement phase, natural language based specification is formalized to NuSCR specification. *NuSRS* also has simple function to check completeness and consistency of NuSCR specification. *NuSRS* generates NuSCR information in an XML format file.

4.2 NuSCRtoFBD

FBD is graphical language for Programmable Logic Controller (PLC) programming defined by International Electrotechnical Commission [9]. The tools which support PLC based system, can automatically generate executable machine code from FBD. *NuSCRtoFBD* automatically translate from NuSCR specification into FBD program. It generates result FBD as a PLCopen standard XML format [10] file.

4.3 FBDtoVerilog

Verilog is a programming language widely used for modeling hardware behavior, and also a front-end of the VIS formal verification tool [11]. In order to apply formal verification by using VIS, FBD programs should be translated into Verilog code. *FBDtoVerilog* is tool for supporting translation from FBD programs into Verilog code. It reads PLCopen standard XML format of FBD, and automatically translates into Verilog code.

3. Conclusions

In this paper, we introduce *NuDE* development environment for safety-critical software of NPP. *NuDE* supports formal methods based techniques. *NuDE* integrated with various tools including *NuSRS*, *NuSCRtoFBD*, and *FBDtoVerilog*. In order to integrate these tools gracefully, we considered IE approach by porting to Eclipse plug-ins from existing independent tools. Through *NuDE*, engineers can reduce the time and develop safety-critical software more effectively.

ACKNOWLEDGEMENT

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the Development of Performance Improvement Technology for Engineering Tool of Safety PLC (Programmable Logic Controller) program supervised by the KETEP (Korea Institute of Energy Technology Evaluation and Planning). (KETEP-2010-T1001-01038)

REFERENCES

[1] I. Sommerville. Software Engineering 9th Edition. Pearson Addison-Wesley, pp.299-302, 2010.
[2] IEEE, IEEE 1012 Standard for Software Verification and Validation, an American National Standard, 2004.
[3] J. Yoo, E. Jee, S. Cha, Formal Modeling and Verification of Safety-Critical Software, IEEE Software, Vol.26, No.3, pp.42-49, 2009.
[4] Eclipse IDE Project. http://www.eclipse.org
[5] J. Cho, J. Yoo, S. Cha, NuEditor – A Tool Suite for Specification and Verification of NuSCR, In proceeding of Second ACIS International Conference on Software Engineering Research, Management and Applications, pp.298-304, 2004.
[6] J. Yoo, S. Cha, C. H. Kim, D. Y. Song, Synthesis of FBD-based PLC design from NuSCR formal specification, Reliability Engineering and System Safety, Vol.87, No.2, pp.287-294, 2005.
[7] J. Yoo, J.-H. Lee, S. Jeong, S. Cha, FBDtoVerilog: A Vendor-Independent Translation from FBDs into Verilog Programs, The 23th International Conference on Software Engineering and Knowledge Engineering, pp.48-51, 2011.
[8] J. Yoo, S. T. Kim, S. Cha, J.-S. Lee, H. S. Son, A Formal Software Requirements Specification Method for Digital Nuclear Plants Protection systems, Journal of Systems and Software, Vol.74, No.1, pp.73-83, 2005.
[9] IEC (International Electrotechnical Commission), International standard for programmable controller: Programming languages: Part 3 (IEC 61131-3), 1993.
[10] PLCopen for efficiency in automation, http://www.plcopen.org.
[11] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. A. Edwards, S. P. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy, T. Villa, VIS: A system for verification and synthesis, In the 8th International Conference on Computer Aided Verification, pp.428-432, 1996.