Transactions of the Korean Nuclear Society Autumn Meeting
Busan, Korea, October 27-28, 2005

Evaluation of Model Driven Development of Safety Critical Software in the Nuclear Power Plant I&C system

Jae-Cheon Jung, Hoon-Seon Chang, Young Woo Chang, Jae-Hack Kim, Se-Do, Sohn
Korea Power Engineering Company Inc.,150 Dukjin-dong, Yuseong-gu, Daejeon, Korea

1. Introduction

The major issues of the safety critical software are formalism and V&V. Implementing these two characteristics in the safety critical software will greatly enhance the quality of software product. The structure based development requires lots of output documents from the requirements phase to the testing phase. The requirements analysis phase is open omitted. According to the Standish group report in 2001, 49% of software project is cancelled before completion or never implemented. In addition, 23% is completed and become operational, but over-budget, over the time estimation, and with fewer features and functions than initially specified [1]. They identified ten success factors. Among them, firm basic requirements and formal methods are technically achievable factors while the remaining eight are management related.

Misunderstanding of requirements due to lack of communication between the design engineer and verification engineer causes unexpected result such as functionality error of system.

Safety critical software shall comply with such characteristics as; modularity, simplicity, minimizing the sub-routine, and excluding the interrupt routine. In addition, the crosslink fault and erroneous function shall be eliminated. The easiness of repairing work after the installation shall be achieved as well [2].

In consideration of the above issues, we evaluate the model driven development (MDD) methods for nuclear I&C systems software. For qualitative analysis, the unified modeling language (UML), functional block language (FBL) and the safety critical application environment (SCADE) are tested for the above characteristics.

2. Formal modeling characteristics of MDD

MDD relies on the use of unambiguous formalisms for specifying systems and brings techniques and tools together to fulfill the objectives of the software process model. MDD is equipped with a specification language using formal semantics and a code generator.

Real-time software for the nuclear I&C system runs on a predefined cycle. A software component starts reacting after an event or a clock by reading its inputs, then performs then the computations according to these inputs and its internal state, and produces outputs [3].

Formal verification increases the reliability and quality of designs by checking their safety requirements. Additionally, automatic code generation ensures that what is verified on the model is verified on the source code [4].

For validation of the generated code, the code review and module test is performed.

3. Selection of MDD based software engineering tool for Nuclear I&C system

For evaluating the formal modeling and verification characteristics of three MDD tools, the safety shut down system (SDS) for nuclear power plant is implemented in this work. SDS consists of 26 functional modules. Among them, 21 functional modules are modeled by SCADE. The remaining modules are not implemented because those modules has hardware related functions such as analog input/output and digital contact.

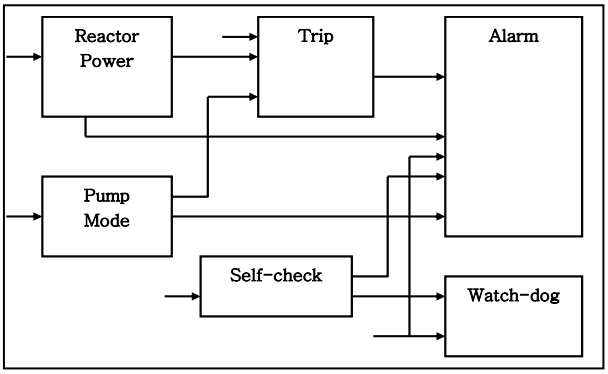

Figure 1. Overall Scheme of Safety Shutdown System implemented by MDD method

Figure 1 shows the overall scheme of SDS for evaluation. It consists of reactor power modules, pump mode modules, trip modules and alarm modules. Individual 21 modules are modeled and then integrated to the system.

Table 1 gives the comparison of time-consumption for modeling and coding results from UML, FBL and SCADE. As is shown, the SCADE needs shorter period than other methods to develop the same software. Using SCADE does not require extra development activities for code generation and code verification. In addition, SCADE can be implemented on most of operating system that compiles ANSY-C language independent of target hardware while FBL can only be used on dedicated hardware system.

The testing activities after code generation can be eliminated in SCADE because all the verification and validation activities can be shifted to the model specification phase.

Table 1 comparison of time-consumption for modeling and coding of MDD methods

Development	Modeling and Coding Period	Code Format
UML	25days	Visual C++
FBL	15days	Dedicated system Code
SCADE	7days	ANSY-C

3. Implementation of safety critical software using SCADE

Figure 2 shows example of the modeled module of calibrated log power (LoCalib Modules). In this case, three macros such as the irrational(IRRAT), changeover switch (UMS), and linear interporator(FKG) are used. SCADE uses both data flow diagram and state machine for system modeling.

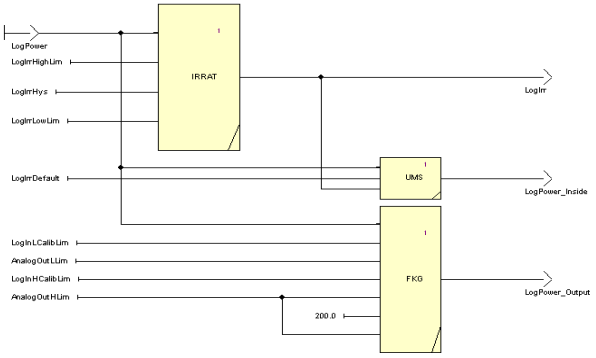

Figure 2. Calibrated log power module modeling

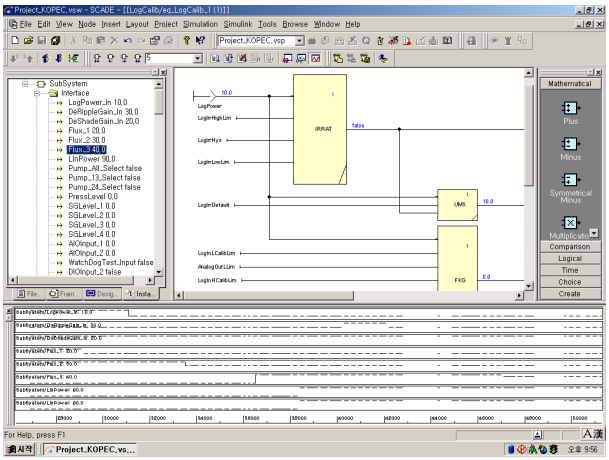

Figure 3. Simulation windows of SCADE

Figure 3 shows the simulation windows of SCADE. Individual models are tested concurrently in order to verify that the specification is correctly implemented to the model. After the simulation, the design verification follows and then the source codes are automatically generated.

Figure 4 shows the execution results after installation on the target system. The target system uses Vxworks, a real-time operating system. The generated codes are executing as expected.

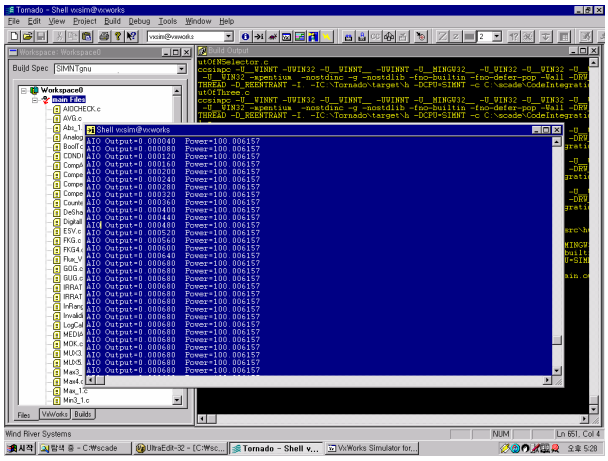

Figure 4. Compile and Execution Result in the Target Real Time System

4. Conclusion

For evaluating the formal modeling and verification characteristics of three MDD tools, main functions of the safety shut down system (SDS) for nuclear power plant is implemented. SCADE needs shorter period than other methods to develop the same software. It does not require extra development activities for code generation and code verification.

Graphical modeling formalism of SCADE benefits from deterministic formal semantics, allowing the deviation of a mathematical model from design. The same deterministic model is then used for automatic code generation and formal verification. This approach can simplify the verification and validation activities.

REFERENCES

[1] Standish Group International, Inc., “Extreme Chaos”, 2001

[2] H. S. Chang, J.C. Jung, J. H. Kim, H. B. Kim, “Requirements analysis of the Safety Critical Software Implementation for the Nuclear Power Plants”, ISOFIC, 2005

[3] Amar Bouali and Bernard Dion, “Formal Verification for Model-Based Development”, SAE World Congress, 2003

[4] Amar Bouali and Bernard Dion, Kosuke Konishi, “Using Formal Verification in Real-time Embedded Software Development”, JSAE Annual Congress, 2005