Formal Modeling and Verification of Safety-Critical Software

Junbeom Yoo, *Konkuk University*

Eunkyoung Jee, *Korea Advanced Institute of Science and Technology*

Sungdeok Cha, *Korea University*

A formal-methods-based process for developing safety-critical software supports development, verification and validation, and safety analysis and has proven to be effective and easy to apply.

Rigorous quality demonstration is important when developing safety-critical software such as a nuclear power plant's reactor protection system (RPS). Although stakeholders strongly recommend using formal modeling and verification, domain experts often reject such methods because the candidate techniques are overabundant, the notations appear complex, the tools often work only in isolation, and the output is frequently too difficult for domain experts to understand and to extract meaningful information.

To overcome such obstacles, we developed a formal-methods-based process that supports development, verification, and safety analysis. We also developed CASE tools to let nuclear engineers apply formal methods without having to know the underlying formalism in depth. In this article, we describe more than seven years' experience working with nuclear engineers in developing RPS software and applying formal methods. Nuclear engineers and regulatory personnel found the process effective and easy to apply with our integrated tool support.

Developing a Digital Control System

When developing and verifying safety-critical software, formal methods are important for increasing safety assurance and demonstrating compliance with strict regulations. In 2001, the Korean Nuclear Instrumentation and Control System consortium (KNICS; www.knics.re.kr) began developing a digital control system for the APR-1400 reactor. At the project's start, project managers made two decisions that strongly influenced our process:

- When developing safety-critical components such as an RPS, we would use formal methods whenever it was practical to do so.
- Software development would be based on the programmable logic controller (PLC), using function block diagram (FBD) as the implementation language.

As a software engineering research group in computer science, we began working with nuclear engineers to produce a formal requirements specification, develop necessary CASE tools, and conduct formal verification during software development. Figure 1 describes the overall process we developed, which covers three essential aspects of safety-critical software: development, formal verification, and safety demonstration.

0740-7459/09/$25.00 © 2009 IEEE