http://dx.doi.org/10.5516/NET.04.2012.085

A BEHAVIOR-PRESERVING TRANSLATION FROM FBD DESIGN TO C IMPLEMENTATION FOR REACTOR PROTECTION SYSTEM SOFTWARE

JUNBEOM YOO[1*], EUI-SUB KIM[1], and JANG-SOO LEE[2]
[1] Konkuk University, Division of Computer Science and Engineering
1 Hwayang-dong, Gwangjin-gu, Seoul, 143-701, Republic of Korea
[2] Korea Atomic Energy Research Institute, Man-Machine Interface System Team
989-111 Deadeok-daero Yuseong-gu, Daejeon, 305-353, Republic of Korea
[*]Corresponding author. E-mail : jbyoo@konkuk.ac.kr

Received November 30, 2012
Accepted for Publication February 12, 2013

Software safety for nuclear reactor protection systems (RPSs) is the most important requirement for the obtainment of permission for operation and export from government authorities, which is why it should be managed with well-experienced software development processes. The RPS software is typically modeled with function block diagrams (FBDs) in the design phase, and then mechanically translated into C programs in the implementation phase, which is finally compiled into executable machine codes and loaded on RPS hardware - PLC (Programmable Logic Controller). Whereas C Compilers are fully-verified COTS (Commercial Off-The-Shelf) software, translators from FBDs to C programs are provided by PLC vendors. Long-term experience, experiments and simulations have validated their correctness and function safety. This paper proposes a behavior-preserving translation from FBD design to C implementation for RPS software. It includes two sets of translation algorithms and rules as well as a prototype translator. We used an example of RPS software in a Korean nuclear power plant to demonstrate the correctness and effectiveness of the proposed translation.

KEYWORDS : Behavior-Preserving Translation, Programmable Logic Controller, Translator, Function Block Diagram, C Program

1. INTRODUCTION

Safety [1] is an important property for nuclear power plants in order to obtain permission from government authorities for their operation and possible export of power plant construction technology. As the nuclear reactor protection system (RPS) makes decisions for emergent reactor shutdown, RPS software should be verified throughout the entire software development life cycle (SDLC). Recent commercial digital I&Cs (Instrumentation & Controls) use a safe-level PLC (Programmable Logic Controller) as a common hardware platform for RPS, *e.g.*, Shin Ulchin 1/2 NPPs in Korea. The RPS software is first modeled with IEC-61131-3 FBD (Function Block Diagram) [2] in the design phase. In implementation, the FBD programs are translated into C programs and then compiled into executable machine code for RPS hardware - PLC. Compiler expert companies typically provide C compilers in which functional correctness is thoroughly verified and demonstrated. Translators from FBDs to C programs are usually developed by PLC vendors. They should sufficiently demonstrate correctness and functional safety [3] of the so-called '*FBD-to-C*' translator.

Vendors such as *AREVA*[1], *invensys*[2] and *POSCO ICT*[3] have provided PLCs and software engineering tool-sets. '*SPACE*' [4] is a software engineering tool-set for *AREVA*'s PLC '*TELEPERM XS*' [5]. It stores FBD programs into a database '*INGRES*' and generates ANSI C programs for code-based testing and simulation ('*TXS SIVAT*' [6]). *ISTec GmbH*[4] has also developed a reverse engineering tool '*RETRANS*' [7] for checking the consistency between FBD programs and generated C programs. The mechanical translator in '*SPACE*' has been validated in such ways, and the software engineering tool-sets have been used successfully for more than a decade. It is worth noting that '*SPACE*' does not use a common C translator for '*RETRANS*' and executable PLC code generation. ('*TXS SIVAT*' uses two ones for different use.) PLCs of *invensys* have also been widely used. '*TriStation 1131*' [8] is its software engineering tool-set. It provides enhanced emulation-based testing and real-time simulation of FBDs, but does not include a translator into C programs.

KNICS (Korea Nuclear Instrumentation and Control System R&D Center) project [9] and *POSCO ICT* in Korea have recently developed a safety-level PLC '*POSAFE-Q*'

[1] *AREVA* (http://www.areva.com)
[2] *invensys* (http://iom.invensys.com)
[3] *PONU Tech* (http://www.ponu-tech.co.kr) for PLC segment was split from *POSCO ICT* (http://www.poscoict.co.kr)
[4] *ISTec GmbH* (http://www.istec.de)

and its software engineering tool-set '*pSET*' [10]. The tool-set provides a graphical editor for FBD and LD (Ladder Diagram) programming languages [2], and also automatically generates the ANSI C program. However, sufficient demonstration of the correctness and functional safety of the so-called '*FBD-to-C*' translator is still in progress as it is considered to be one of the most critical obstacles that must overcome in order to obtain permission for the export of the new Korean nuclear power plant [11] as a whole, *i.e.*, including control software - I&C (Instrumentation & Control).

This paper presents a technique for the development of an '*FBD-to-C*' translator, guaranteeing their fundamental behavioral equivalence without the aid of simulation and testing techniques. We propose two sets of algorithms and rules, which translate FBDs in the design phase into ANSI C programs in the implementation phase. The proposed translations use only a restricted subset of C programming language and do not need a full-scale verification, typically used in the discipline of programming [12]. We also implemented a prototype of the '*FBD-to-C*' translator and performed a case study with an example of RPS recently developed in Korea. It is not the final version of the RPS software, against which government authorities have been evaluating for years, but a preliminary one developed for the purpose of diversity and prototyping. We generated FBD programs mechanically from formal requirement specifications [13,14] as explained in [15]. The example, however, is sufficient to demonstrate that the proposed translation techniques work on all types of shutdown logics to which the RPS should give consideration.

The paper is organized as follows: Section 2 gives an introduction to a typical development process for RPS software and several techniques for validating '*FBD-to-C*' translators. We focus on *AREVA*, France and *POSCO ICT*, Korea. In section 3, we briefly introduce formal definitions of FBDs, which are pertinent to our discussion. Section 4 explains translation algorithms and rules from FBDs into a subset of ANSI C programs. In order to aid understanding, we explain the rules with an example, a basic shutdown logic; '*fixed set-point rising trip*.' Section 5 presents the result of the case study, encompassing all shutdown logics of a preliminary version of Korean APR-1400 RPS. It also demonstrates that our translation can apply to all logics of RPSs efficiently and sufficiently. Finally, Section 6 concludes the paper and provides remarks on future research extension and direction.

2. THE RPS SOFTWARE DEVELOPMENT PROCESS

RPS (Reactor Protection System) is a real-time embedded system, implemented on the hardware - PLC (Programmable Logic Controller). The RPS software is designed in FBD/LD languages and then translated into C programs which will be compiled and loaded on PLCs. Fig.1 explains a typical software development process for RPSs as well as the techniques used for validating the '*FBD-to-C*' translator.

The upper part describes a typical software development process as a waterfall model [16]. The SRS (Software Requirements Specification) is written in natural languages or formal specification languages [15,17,18]. Experts on PLC programming languages then translate the requirement specifications manually into design models programmed in FBD or LD. PLC vendors provide their own automatic translators from FBD/LD programs into ANSI C programs, while typically using COTS (Commercial Off-the-Shelf) software such as '*TMS320C55x*' of *Texas Instruments* [19] for C compilers. The COTS compilers were well verified and sufficiently certified to be used for implementing the RPS software without additional effort.

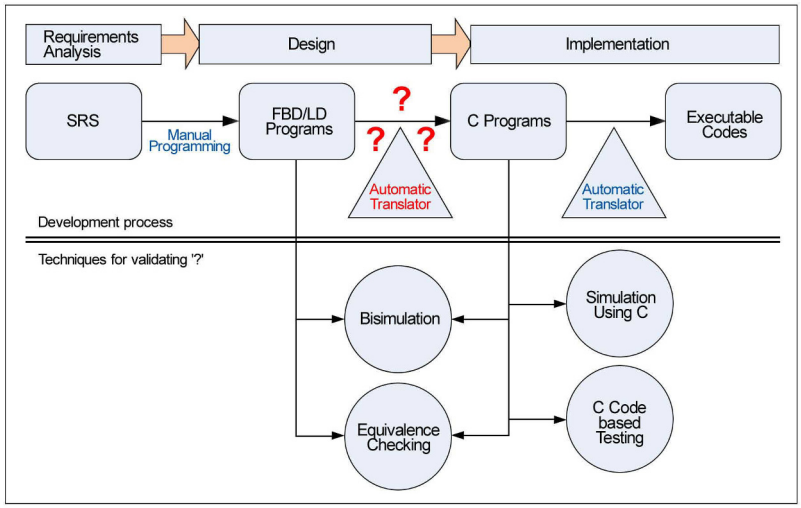

Fig. 1. A Software Development Process for RPS