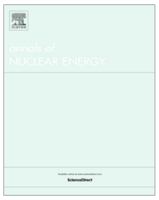
Annals of Nuclear Energy 103 (2017) 454–469

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene



A PLC platform-independent structural analysis on FBD programs for digital reactor protection systems ☆

Sejin Jung [a], Junbeom Yoo [a,*], Young-Jun Lee [b]

[a] Konkuk University, Republic of Korea
[b] Korea Atomic Energy Research Institute, Republic of Korea

ARTICLE INFO

Article history:
Received 29 March 2016
Received in revised form 3 February 2017
Accepted 5 February 2017

Keywords:
Structural analysis
Function block diagram
PLC
Guidelines

ABSTRACT

FBD (function block diagram) has been widely used to implement safety-critical software for PLC (programmable logic controller)-based digital nuclear reactor protection systems. The software should be developed strictly in accordance with safety programming guidelines such as NUREG/CR-6463. Software engineering tools of PLC vendors enable us to present structural analyses using FBD programs, but specific rules pertaining to the guidelines are enclosed within the commercial tools, and specific links to the guidelines are not clearly communicated. This paper proposes a set of rules on the structure of FBD programs in accordance with guidelines, and we develop an automatic analysis tool for FBD programs written in the PLCopen TC6 format. With the proposed tool, any FBD program that is transformed into an open format can be analyzed the PLC platform-independently. We consider a case study on FBD programs obtained from a preliminary version of a Korean nuclear power plant, and we demonstrate the effectiveness and potential of the proposed rules and analysis tool.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

PLCs (programmable logic controllers) have been widely used in the development of embedded controllers of various safety-critical systems. The software implemented with PLCs is typically programmed with PLC programming languages such as FBD (function block diagram), and then software engineering tools of PLC vendors synthesize the programs into PLC-executable codes mechanically. All issues related to structural correctness and the safety of the FBD programs need to be addressed in order to proceed with the mechanical synthesis. Software engineering tools developed by PLC vendors can check these issues strictly, but vendor-dependently and SW tool-internally.

RPS (Reactor Protection System) in nuclear power plants is implemented using PLCs, and the software used is often programmed with FBD. Because these systems are safety-critical systems, a hierarchy of standards, regulations, and guidelines on the structural quality (i.e., correctness and safety) of software programs should be satisfied to obtain operational approval of government authorities. IEC 61131-3 (IEC, 2013), IEC 61508 (IEC, 1997), and NUREG/CR-6463 (NRC, 1997) are some examples of such standards. NUREG/CR-6463 is top-most programming guideline for software development, and all software engineering tools developed by PLC vendors apply it in the analysis of FBD programs. In this paper, we focus on the problem whereby commercial tools perform the structural analysis well, but the exact correspondence (or relation) to upper rules and guidelines is neither clear nor opened.

In this paper, we propose a set of specific rules regarding the structure of FBD programs in accordance with the guidelines of higher levels. We can argue direct relations from one rule/guideline to upper ones. We also developed an automatic analysis tool "*FBD Checker*" for FBD programs in the PLCopen TC6 format (PLCopen XML schema Ver. 2.0, 2008). Any FBD program that is written or transformed into the open format can be analyzed the PLC platform[1]-independently. In other words, we do not have to use the software engineering tools of specific PLC vendors. We also performed case studies of structural analyses on FBD programs that were sampled from preliminary versions of Korean nuclear power plants. The results show the effectiveness and potential of the proposed rule sets and analysis tool.

The paper is organized as follows. In Section 2, we discuss background information such as FBD programming. Section 2 also

☆ This paper was originally published in *Korean Nuclear Society Autumn Meeting* 2014 (Jung et al., 2014).
* Corresponding author.
E-mail address: jbyoo@konkuk.ac.kr (J. Yoo).

[1] This paper uses the term '*PLC platform*' to indicate the pair (a PLC software engineering tool, a target PLC) such as (*pSET*, *POSAFE-Q PLC*) and (*SIMATIC-Manager*, SIMATIC Controller).

http://dx.doi.org/10.1016/j.anucene.2017.02.006
0306-4549/© 2017 Elsevier Ltd. All rights reserved.

describes a hierarchy of standards, rules, and guidelines for FBD programming, which are relevant to our discussion, i.e., developing PLC software in nuclear reactor protection systems. In Section 3, we explain detailed sets of guidelines that are proposed in this paper, along with examples, and in Section 4, we introduce the automatic structure analyzer - FBD Checker. We explain the case study in Section 5, while in Section 6, we conclude the paper and give remarks on our future research direction.

2. Background

2.1. FBD programming

FBD is one of the five PLC programming languages defined by the IEC 61131-3 standard (IEC, 2013), and it is the most widely used language to implement PLC-based safety-critical systems in nuclear power plants. FBD is a data-flow-based language that consists of function blocks that connect with each other. FBD programming is the process of connecting blocks to other blocks sequentially in order to produce appropriate outputs. Fig. 1 is an example of an FBD program, which is a part of '*fixed set-point rising trip*.' As we will explain in Section 5, this FBD program was originally an *NuSCR (nuclear software cost reduction)* (Yoo et al., 2005) formal requirements specification (KAERI, 2003), and the *NuDE (nuclear development environment)* framework (Yoo et al., 2014a; Yoo et al., 2014b; Kim et al., accepted) transformed it into a behaviorally equivalent FBD program.

FBD program in Fig. 1 consists of five function blocks, and it has a set of interconnections according to a predefined sequential execution order such as (27)–(31), which is labeled in Fig. 1. For example, the first function block that is executed is `LT_INT(27)`, while the last one is `AND_BOOL(31)`. `LT_INT` is a function block that calculates the logical '<' with two decimal integer inputs, and other function blocks can be understood in a similar way. The last function block `AND_BOOL` produces an output *TRIP*, which indicates a shutdown signal for nuclear reactors.

Fig. 2 shows a typical software development process for PLCs, used to develop safety-grade digital I&Cs. We first wrote the SRS (Software Requirements Specification) using in natural languages, and we then manually modeled the design specification using PLC programming languages such as FBD or LD. As commercial PLC vendors provide software engineering tools[2] to support mechanical translation from FBD/LD programs into C and executable codes for PLCs, most manual software programming will finish at the design phase. Structural analysis on FBD programs is also performed by PLC software engineering tools.

2.2. FBD Programming guidelines for safety systems in nuclear power plants

Fig. 3 summarizes FBD programming guidelines for safety systems in nuclear power plants, which are pertinent to our discussion - '*structural analysis on FBD programs*.' Below, the IEC 61131 Part 3 (IEC, 2013) defines 5 PLC programming languages, e.g., FBD, LD (Ladder Diagram) and ST (Structured Text), while Part 8 (IEC, 1993) provides basic programming guidelines to be followed with PLC programming languages. They define the FBD programming language and provide guidelines regarding how to program with FBD for PLC-based systems of general-purpose.

Based on the standards, the technical report of PLCopen TC5 (Safety Software Technical Specification, 2006; Safety Software Technical Specification, 2008) provides FBD programming guidelines for safety-critical systems. It suggests safe data types, which contain additional information for the safety status and level, as well as safe function blocks. PLCopen TC6 (PLCopen XML schema Ver. 2.0, 2008) also defines an open XML format for FBDs to enable the exchange of FBD programs with others, and this is because FBD programs produced by commercial software engineering tools of PLC vendors are not compatible with others. In this paper, we use the XML format to perform a structural analysis on FBD programs, vendor and tools (*i.e.,* the PLC platform-independently).

NUREG/CR-6463 (NRC, 1997; NRC, 1997) provides guidelines on software programming languages for nuclear power plant safety systems, as defined by the NRC (Nuclear Regulatory Commission) (NRC, 2015). It provides the following high-level languages, *e.g.,* Ada, C/C++, LD, FBD, Sequential Function Charts (SFC), Pascal, and PL/M. Further, it consists of 4 high-level categories such as *reliability*, *robustness*, *traceability*, and *maintainability*. The guidelines with respect to *reliability* are to improve the dependability and guarantee correctness, while the guidelines with respect to *maintainability* increase the readability and decrease the complexity. The *robustness* contains exception handling, and so on. The category of *reliability* also consists of 3 sub-chapters, namely *predictability of memory utilization*, *control flow*, and *timing*, and others also have several sub-chapters generically.

PLC vendors have provided safety-level PLCs and their own software engineering tools for developing safety-critical systems in nuclear power plants, as shown in Fig. 3. The commercial tools contain internal structural analysis facilities, which are internally referred to as the NUREG/CR-6463 guidelines. However, rules on FBD structures and specific mapping from the rules to the higher guidelines are not made public. While basic guidelines (SIEMENS PLC Control Systems, 2015; Invensys, 2006) with respect to how to program with function blocks have been publicized, specific rules in accordance with the higher guidelines have not been publicized.

2.3. Related work

To the best of our knowledge, few studies have focused on structural analysis using FBD programs. Lee et al. (2014) proposes 5 categories of FBD programming guidelines for safety-critical systems, such as data type, variable initialization, usage of variable, execution control, and explicit ordering. However, it does not clearly correspond to the top-most guideline, NUREG/CR-6463, whose target applications are safety systems in nuclear power plants. de Mario (2008) proposes 9 kinds of restrictions that should be followed to program with the IEC 61131-3 programming languages for high-integrity applications. Type safety, memory access, global variables, and conversion are some examples of the proposed restriction categories. As the restrictions are proposed for all PLC programming languages, e.g., LD and SFC, we need to select appropriate ones for FBD. The restrictions also need to be refined in detail to enable their use as rules for rule checking. For example, it deals with conversion between integers only.

Several researches that focus on static analysis, not rule checking on FBD programs, are as follows: Prahofer et al. (2012) provides 7 issues for static analysis on the IEC 61131-3 languages. Program complexity, unreachable codes, and performance problems are some examples. It also includes areas that are related to rule checking, such as naming convention. *Codesys* (CODESYS, 2015) is a programming tool of IEC 61131-3 programming languages that provides a static analysis on FBD programs. Using a technical data sheet (CODESYS, 2015), it also provides a list of subjects that includes useless declaration, detection of unreachable code, and naming convention. Fig. 4 depicts the above studies and our proposed approach – '*FBD Checker*' from the perspective of structural analysis on FBD programs.

[2] *e.g., 'TriStation 1131'* of *Invensys* (Invensys, 2015), *'SIMATIC-Manager'* of *Siemens* (SIEMENS, 2015), *'pSET'* of *PONU-Tech* (PONU-Tech, 2015; Cho et al., 2007), and *'SPACE'* of *AREVA* (AVERA, 2015).

Download English Version:

https://daneshyari.com/en/article/5475241

Download Persian Version:

https://daneshyari.com/article/5475241

Daneshyari.com