LETTER

An Empirical Evaluation of Coverage Criteria for FBD Simulation Using Mutation Analysis

Dong-Ah LEE¹, Eui-Sub KIM¹, Nonmembers, and Junbeom YOO²[a], Member

SUMMARY Two structural coverage criteria, toggle coverage and modified condition/decision coverage, for FBD (Function Block Diagram) simulation are proposed in the previous study. This paper empirically evaluates how effective the coverage criteria are to detect faults in an FBD program using the mutation analysis.

key words: coverage adequacy criteria, FBD simulation, mutation analysis

1. Introduction

Software testing is one of indispensable activities in the software development process. Software testing methods are traditionally divided into functional (black-box) testing and structural (white-box) testing, in which test cases are derived from program specifications and from the structure of programs, respectively. Functional testing verifies the functional correctness of software in any step of the software development process. Structural testing not only verifies the functional correctness but also measures coverage which means what percent of code has been exercised when a test suite runs. One or more structural coverage criteria, such as statement coverage, branch coverage, and condition coverage, are used to measure the coverage.

FBD (function block diagram) is a commonly used programming language to develop software for PLC (programmable logic controller). Safety-critical systems often use the FBD to design software for digital I&Cs (instrumentation and control system). For example, the KNICS (Korea Nuclear Instrumentation and Control System) project implemented trip (shutdown) logics of a BP (bistable processor) for RPS (reactor protection systems). Testing FBD software often performs simulation-based testing for functional verification. The previous study proposed two sets of structural coverage criteria for simulation scenarios of FBD, toggle coverage (TC) and modified condition/decision coverage (MC/DC), also known as simulation coverages. The simulation coverages are similar to structural coverage criteria of software testing, however they are for structural elements of an FBD program and the software testing is for source codes. This paper empirically evaluates the effectiveness of the simulation coverages using mutation analysis. Software requires rigorous quality when developing safety-critical systems such as digital I&Cs in nuclear power plants (NPPs). Simulation verifies functional correctness of the software written in FBD. The simulation requires strong criteria to improve confidence in thoroughness. The mutation analysis is a fault-based software analysis technique to measure the adequacy of a test suite or the effectiveness of an adequacy criterion. The analysis seeds artificial faults (mutations) into an FBD program, then the simulation with a set of scenarios tries to detect the faults in the FBD program having one of the faults (mutants). If the scenarios achieving a higher percent of coverage finds more mutants than ones with a lower percent, the coverage criterion is effective to detect the faults.

The analysis uses trip (shutdown) logic programs of BP, which is a part of the RPS developed in the KNICS project. We generated three types of simulation scenarios, random, guided, and manual scenarios. Results of the mutation analysis show that simulation scenarios achieving a higher percent of both coverages (TC and MC/DC) detect more mutants than ones achieving a lower percent. In other words, the both coverage criteria are suitable for use as a measure of whether simulation scenarios are sufficient to detect faults in an FBD program.

The remaining part of the paper proceeds as follows: Sect. 2 briefly introduces the structural coverage criteria for FBD simulation and mutation analysis. Section 3 gives a full explanation of research questions and evaluation process and Sect. 4 explain analysis results of the evaluation. Finally, Sect. 5 concludes the paper and provides remarks on future research.

2. Background

2.1 Structural Coverage Criteria for FBD Simulation

FBD, one of the five standard PLC programming languages, is a commonly used graphical language to develop software for safety-critical systems. For example, the KNICS project used FBD to implement control software of NPPs. The FBD program in Fig. 1 is a simplified trip (shutdown) logic. It has 5 blocks (2 AND, 1 OR blocks), 7 inputs (3 integer (I) and 4 boolean (B) inputs), and 1 boolean output (TRIP). RNG_MIN and RNG_MAX are constants fixed with 10 and 20,000 respectively.

Simulation verifies that an FBD program is function-
ally correct. Thoroughness, quality, or effectiveness of simulation scenarios are important to increase confidence of the simulation. Structural coverage criteria for FBD simulation [2, 7] improve and refine the simulation scenarios quantitatively.

(1) Toggle coverage (TC)

TC measures how many boolean outputs of blocks in an FBD program are changed from a value of zero to one (0-to-1) and back from one to zero (1-to-0) during simulation. An output is fully covered, i.e. 100% TC, when it toggles back and forth at least once. For example, the output of the block LT_INT in Fig. 1 is fully covered, when the PV_OUT has 5, 15, and 5 sequentially. Simulation of an FBD program uses a set of simulation scenarios, and (TC) measures all toggles during the simulation with massive scenarios. For instance, the FBD program in Fig. 1 has 10 possible toggles which the five blocks have two possible toggles. If a set of simulation scenario toggles all possible ones, then simulation using the scenarios achieves 100% TC.

(2) Modified condition/decision coverage (MC/DC)

MC/DC measures how many important combinations of blocks in an FBD program simulation covers. The important combinations means sets of inputs for a condition of a block which independently affects an output of the block. For example, combinations of inputs of (3) AND_BOOL block has four possible input sets ((0,0), (0,1), (1,0), (1,1)), however only three combinations (i.e., (0,1), (1,0), (1,1)) are important combinations. MC/DC counts all the important combinations executed with respect to a set of simulation scenarios along the same way as TC.

2.2 Mutation Analysis

Mutation analysis, which is often called mutation testing for software testing, is one of software analysis techniques to measure the adequacy of a test suite or the effectiveness of a test adequacy criterion. Research activities about techniques and tools of mutation analysis are increasing, and applicability is getting widespread [3, 8]. A mutant is a modified version of an original program, which has an artificial fault. The mutation analysis tries to detect the mutant using test suites—distinguish the behavior of the mutant from that of the original one—and evaluates the adequacy of the test suites. If a test suite is adequate for an test criterion and detects mutants as much as the adequacy, the test criterion is effective to assess quality of the test suite. Using mutation analysis, this paper analyzes coverage criteria for FBD simulation proposed in the previous work [2].

Shin [12] [13] analyzed FBD test coverage criteria using mutation analysis. The criteria measure coverages about testing of FBD programs. Each test case independently executes an FBD program, which means that the test cases do have single scan cycle. The criteria in this paper are for simulation-based testing of FBD programs, however. The simulation executes an FBD with the use of simulation scenarios which have multiple scan cycles. It is worth noting that it is necessary to execute FBD programs with multiple scan cycles to verify the function correctness because PLC programs are executed in a permanent loop.

3. Empirical Design

3.1 Research Questions and Subjects

This paper investigates the following research questions:

- **RQ1**: How effective is TC for FBD simulation in fault detection?
- **RQ2**: How effective is MC/DC for FBD simulation in fault detection?

To answer the questions, we designed our experiments as described in Fig. 2. We generated three types of simulation scenarios, random (Sr), guided (Sg), and manual scenarios (Sm), for an original FBD program using an automated...
tool, *FBDScaGen*+ [9]. The processing value, which is a name of processing data for the FBD program, has random values in the $S_r$. On the other hand, the value in the $S_y$ has guided values, such as an increase or decrease. The $S_m$ is manually generated by a domain expert. We simulated an original FBD program with each set of simulation scenarios and measured $TC$ and $MC/DC$ using *FBDCover* [7]. Meanwhile, we applied mutation operators in Sect. 3.2 to the original FBD program in order to generate a number of mutants. We also simulated the mutants, i.e., faulty FBD programs, with the three sets of simulation scenarios and measured how many mutants the scenarios detect. Finally, we analyze them in order to answer the research questions.

The experiment uses FBD programs [5] for the second phase of KNICS APR-1400 RPS BP [6] as an original FBD program. It was excerpted from an almost (but, not officially final version) commercial NPP in operation. The BP consists of 18 shutdown logics written in FBD, but we only use 5 representative trip programs, ‘fixed set-point falling trip’ (FFT), ‘variable set-point falling trip’ (VFT), ‘manual reset falling trip’ (MFT), ‘fixed set-point rising trip’ (FRT), and ‘variable set-point rising trip’ (VRT), in this experiment.

### 3.2 Mutant Generation and Mutation-Score Measurement

Mutants should be plausible as faulty programs. In other words, the faults represent mistakes that programmers may make. The mutants are created by seeding such faults following a pattern which is called mutation operators. We defined mutation operators for an FBD program base on earlier research [10]. Table 1 lists five mutations operators. The list includes common mistakes during FBD programming. It does not include faults which tools can identify, however. For example, ‘FBDChecker’ [11] identifies the type mismatch or missing links.

The mutation analysis measures that how much mutants a set of simulation scenarios detect during a simulation, called a mutation-score. The mutation-score, $K$, is described as follows:

$$K = \frac{\text{a number of detected mutants}}{\text{a number of total mutants}} \times 100(\%)$$

We measure the $K$ for each set of simulation scenarios for each program. If a set of simulation scenarios finds all mutants, then the $K$ is 100%. Table 2 indicates the numbers of mutants for each FBD programs we generated.

### 4. Analysis Results

A simulation scenario has 100 execution cycles and a set ($S_r$, $S_y$, $S_m$) includes 1,000 simulation scenarios. Each of the sets reports $TC$, $MC/DC$, and $K$ of the 5 FBD programs individually. $S_r$ achieved 10–15% $TC_{Sr}$ and 40–45% $MC/DC_{Sr}$, while $S_y$ achieved 64–66% $TC_{Sy}$ and 72–74% $MC/DC_{Sy}$. The results of $S_r$ and $S_y$ means that a set of simulation scenarios which is generated by a guidance is more effective to achieve the both coverages than one randomly generated. $S_m$ achieved 85–88% $TC_{Sm}$ and 91–92% $MC/DC_{Sm}$ against all the original FBD programs.

Mutant generation uses only one mutation operator in Table 1 to the original programs in order to generate one mutant. Tens of mutants are generated by the operators for each FBD programs we generated.

### Table 1 Mutation operators for an FBD program

<table>
<thead>
<tr>
<th>ID</th>
<th>Operator</th>
<th>Description</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVR</td>
<td>constant value replacement</td>
<td>replace a constant value C1 by C2</td>
<td>$C1 \neq C2$</td>
</tr>
<tr>
<td>ABR</td>
<td>arithmetic block replacement</td>
<td>replace arithmetic block $\psi$ with arithmetic $\phi$</td>
<td>$\psi_1 \neq \psi_2$</td>
</tr>
<tr>
<td>LBR</td>
<td>logical block replacement</td>
<td>replace logical block $\phi$ with logical block $\phi'$</td>
<td>$\phi_1 \neq \phi_2$</td>
</tr>
<tr>
<td>CBR</td>
<td>comparison block replacement</td>
<td>replace comparison block $\psi$ with comparison block $\phi'$</td>
<td>$\phi_1 \neq \phi_2$</td>
</tr>
<tr>
<td>IVR</td>
<td>input variable replacement</td>
<td>swap an input variable $V_f$ with another</td>
<td>$</td>
</tr>
</tbody>
</table>

### Table 2 Summary of $TC$, $MC/DC$, and $K$ for the three types of simulation scenarios

<table>
<thead>
<tr>
<th>Name of FBD programs</th>
<th>Number of Mutants</th>
<th>Types of simulation scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$S_r$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$TC_{Sr}$ MC/DC$<em>{Sr}$ K$</em>{Sr}$</td>
</tr>
<tr>
<td>FFT</td>
<td>43</td>
<td>11</td>
</tr>
<tr>
<td>VFT</td>
<td>63</td>
<td>15</td>
</tr>
<tr>
<td>MFT</td>
<td>67</td>
<td>15</td>
</tr>
<tr>
<td>FRT</td>
<td>43</td>
<td>12</td>
</tr>
<tr>
<td>VRT</td>
<td>63</td>
<td>14</td>
</tr>
</tbody>
</table>
strong strength. In other words, the higher percent of TC a set of simulation scenarios achieves, the more mutants the set detects.

(2) RQ2: How effective is MC/DC for FBD simulation in fault detection?

The graph (b) in Fig. 3 shows the correlation of \( K \) with the coverage criteria (TC and MC/DC). The value of \( r = 0.9777 \) indicates a positive and linear relationship of strong strength between the two variables. The higher percent of MC/DC in common with TC, a set of simulation scenario achieves, the more faulty program the set finds.

The results give us information that both coverage criteria, TC and MC/DC, are effective to detect a fault in an FBD program. A set of simulation scenario usually achieves the higher percent of MC/DC than TC. Even if a set achieves a low percent of about 10% TC, the set finds more than half of mutants. In order to detect faults sufficiently, however, the set should achieve a sufficiently high percent of both criteria. TC should be over about 87% and MC/DC should be over about 91% to detect over 90% of faults in the experiment.

One of the results indicates that the detection of mutants by CBR is relatively more difficult than others. The mutation-score \( S_T \) for LBR is about 66% and \( S_T \) for CBR is about 25% for FFT. \( S_T \) for LBR is about 89% and \( S_T \) for CBR is about 63%. The reason is that some of mutants by CBR make a slight difference, such as generating \( \text{LE}_\text{INT} \) from \( \text{LT}_\text{INT} \). This means that it is difficult to detect these kinds of fault by simulation-based testing.

5. Conclusion and Future Work

This paper reports empirical evaluations for FBD simulation coverage criteria by mutation analysis. We used 5 representative FBD programs for the evaluation and generated tens of mutants for each programs using mutation operators. The experimental results demonstrated that simulation scenarios which achieve a higher percent of coverages are more effective to detect faults in an FBD program.

The most important limitation lies in the fact that achieving a sufficiently high percent of coverages takes lots of time and effort. It was possible to generate simulation scenarios achieving over 87% TC and 90% MC/DC by domain experts manually. Although FBDScenaGen+ generated a number of simulation scenarios automatically, they only detected under 84% of mutants. We have a plan to improve the scenario generation using a machine learning technique in our future work.

Acknowledgments

This research partially supported by Next-Generation Information Computing Development Program through the National Research Foundation (NRF) of Korea funded by the Ministry of Science, ICT (NRF-2017M3C4A7066479).

References