USING TIMED BASE-CHOICE COVERAGE CRITERION FOR TESTING INDUSTRIAL CONTROL SOFTWARE

HENNING BERGSTRÖM AND EDUARD ENOIU

IWCT 2017
COMBINATORIAL TESTING (CT)

• CT tries to select test input values:
  • the test goal is a combination strategy - test criteria.

• Test Level:
  • System: create inputs on user-level interaction
  • Unit: create inputs for method param. and variables
COMBINATORIAL TESTING (CT)

- CT tries to select test input values:
  - the test goal is a combination strategy - test criteria.

- Test Level:
  - System: create inputs on user-level interfaces
  - Unit: create inputs for method parameters and variables

TEST CRITERIA APPLIED TO UNITS
PROGRAMMABLE LOGIC CONTROLLERS (PLC)

- Are real-time systems
- Found in trains, nuclear power plants, automation
- Run on domain-specific operating systems.
INDUSTRIAL CONTROL SOFTWARE (UNIT) PLC WRITTEN IN IEC 61131-3

Diagram:
- AND gate
- OR gate
- Timer (TON) with 5s delay
- Inputs: IN1, IN2, IN3, IN4
- Outputs: OUT
- Constants: -55, 125
Unit (component) level testing
- A design is used as expected output (test oracle)
- The use of functional testing is mandated
- Some level of code coverage is recommended
BASE CHOICE CRITERION

<table>
<thead>
<tr>
<th># Tests</th>
<th>IN1</th>
<th>IN2</th>
<th>IN3</th>
<th>IN4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

program cycle P=500ms
TIMED BASE-CHOICE CRITERION

1. Create the basic input model
2. Identify the timing constraint
3. Identify a base and time choice test.
4. Create a test suite.

- time choice $T = 6s$
- $(1,1,5,4)$ is fixed for 6s.
- Tests 1 to 7 are fixed for 6s each
CASE STUDY

• Compare timed base-choice with base choice in terms of code coverage and fault detection.
CASE STUDY

METHOD

11 PROGRAMS

BC

TBC

RAND

TESTS

TESTS

TESTS

BRANCH COVERAGE

FAULT DETECTION
MUTATION ANALYSIS

MUTANT OPERATORS

- COMPARISON
- VALUE
- NEGATION
- LOGICAL
- ARITHMETIC
- TIMER

(Yoo et al 2007) (Shin et al. 2012)

IEC 61131-3 PROGRAM

MUTANTS
MUTATION ANALYSIS

FIND TESTS WHICH WHEN EXECUTED ON BOTH ORIGINAL & MUTANT PROGRAMS OUTPUT/STATE IS DIFFERENT
RESULTS

TBC ACHIEVES BETTER FAULT DETECTION SCORES THAN BC OR RAND.

The results show that timed base-choice criterion is useful for designing tests for industrial control software containing timing behavior. The definition of base-choice criterion has been proposed for testing IEC 61131-3 industrial control software. We conducted a case study in which we compared the cost and effectiveness between timed base-choice and base-choice for each of the collected test suites. The use of timed base-choice criterion has been demonstrated to achieve better code coverage and fault detection compared to both base-choice and random test suites. The fault detection scores of TBC test suites are superior to either BC test suites (57% mutation score on average) and random test suites (42% mutation score on average). Hence a test being created using TBC is a good indicator of test effectiveness in terms of fault detection. As shown in Figure 3a, the fault detection scores of TBC test suites are significantly higher than those of BC and Rand test suites. Based on the results, we can conclude that the use of timed base-choice generated test suites is effective for testing IEC 61131-3 industrial control software.

REFERENCES

RESULTS

TBC achieves better decision coverage scores than BC or Rand.

![Box plot showing decision coverage comparison between Rand, BC, and TBC.](image)
In this study we proposed the use of timed base-choice criterion for testing IEC 61131-3 industrial control software. We conducted a case study in which we compared the cost of applying this criterion. The results show that timed base-choice criterion is useful for achieving higher code coverage and fault detection compared to base-choice tests. The use of timed base-choice generated test suites are superior to either BC test suites (57% mutation score on average) and random test suites (42% mutation score on average) and are clearly matching our expectations: TBC test suites show an average mutation score of 84% and are clearly showing an average fault detection scores, code coverage results and the number of test cases comparing the test suite quality. For all programs, as shown in Figure 3a, the fault detection scores of TBC test suites are seen in Figure 3b, the use of TBC achieves on average 85% decision coverage. The answer to question RQ2 regarding the effectiveness in terms of fault detection, we focused on the circle symbols represent outliers.

In Figure 3c, the use of TBC consistently results in significantly more number of tests (with 300% more tests highlighted in Figure 3c, the use of TBC achieves 78% code coverage which is higher than BC (on average 57%) and random test suites (42% mutation score on average) from 1st to 3rd quartile, black middle lines mark the median and the whiskers extend up to 1.5x the inter-quartile range and such as t-wise testing.

We would like to highlight that for testing IEC 61131-3 industrial programs written in the IEC 61131-3 programming languages, the use of timed base-choice criterion needs to be further studied; we need to consider the implications of using multiple base and time choices. In addition, base-choice is only one type of combination strategy such as t-wise testing. Therefore, we would need to evaluate the use of stronger criteria such as t-wise testing.
THE FUTURE

• multiple base and time choices
• evaluate the use of stronger criteria
• Use naturally-occurring faults
• Use other systems from other domains
Using Timed Base-Choice Coverage Criterion for Testing Industrial Control Software

IEC 61131-3 Programs → Timed Base-Choice (TBC) → Base-Choice (BC) → Random (Rand) → Tests → Faulty Programs → Execution Framework → Coverage → Fault Detection

- Fault Detection
- Mutation Score (%)
- Rand
- BC
- TBC

Code Coverage

Fault Detection

BC

TBC