Compare of Formal Analysis and Testing for Verification of Safety-critical Systems: a Case Study

Juan Zhang, Guoqi Li and Xiao Liu

School of reliability and system engineering, Beihang University,

100191 Beijing, China

zhangjuan198804@163.com, gqli@buaa.edu.cn, liuxiaork@hotmail.com

Keywords: formal analysis, testing, verification, safety-critical systems

Abstract. Safety-critical system attracts more attention in recent years. During the development of safety-critical systems, verification plays the most important role and includes many high cost activities. Testing and formal analysis are two mainstream ways for verification. This paper describes new tools and procedures for testing and formal analysis for verification of safety-critical systems. Compare them in detail in a case study. Conclusion and future works are given finally.

Introduction

Safety-critical systems are computer, electronic or electromechanical systems in which failure may have severe consequences such as injury or death of humans. Popular to speak, a safety critical system is designed to lose less than one life per billion hours of operation. In many domains, such as medical instrumentation, railway signaling, air traffic control, onboard systems and etc.. Safety critical systems require the utmost care in their specification and design as well as implementation. Nowadays, Safety-critical systems are usually controlled by embedded computers and software plays a dominant role in their operation.

Software safety verification is to check if the safeties of software satisfy the development requirements according to the contract in the development and using stage. Additionally, to give the advices for inspection, examination, testing, or evaluation works.

Verification of safety-critical software

DO-178C/ED-12C [1] is the current standard for software assurance in the civil aeronautical domain. The verification process of DO-178C includes review/analysis activity and test activities. The progress refer to the system requirements, high-level requirements, software architecture, low-level requirements, source code and executable object code, which is based on the sequence of software development process. Each step of the development process and the objectives should be verified.

Test is used to verify that the executable object is compliant with low-level requirements and high-level requirements or not. Test is always based on the requirements and includes normal range and robustness case. The essence of test verification of DO-178C is requirement coverage and structural coverage of test cases.

Formal method is stated as an effective and challenging one and it will effects the development verification process. Formal methods can be applied to many developments and verification activities required software. Nowadays, the using of formal methods has become relatively mature.

Compare of Formal Analysis and Testing

Formal method has applied widely in the software design and verification stages. The main process of verification using formal methods is review, analysis, test activities and verification and verification activities. Formal analysis can replace the conventional methods of review, analysis, and test for some verification objective. In can also provide guarantees or proofs of software properties and compliance with requirements, all execution cases are taken into account. We can see the function of formal methods is powerful in the verification activities. There are many kinds of formal analysis, but they can typically be classified in three categories: (1) deductive methods, (2) model checking, and (3) abstract interpretation. When use the formal analysis, the results of verification of verification process is complicated. Because guarantee software have been tested enough is by the means of coverage analysis. [2]

In DO-178C, test is the only means envisaged for meeting the verification objectives for executable object code，and it always be the primary means. Formal method was mentioned maybe as the means of test coverage analysis, in some extent, review and analysis activities cannot replace all testing, and the most probably is just a formal analysis method can replace some properties. Test is the basic unit of all engineering discipline, and also an important of the software development. [3]

It seems that the relationship between testing and verification is clear in the theoretical point of view, but it is difficult to understand and many people do not know how to select in practical engineering. In the following section, a case study is given for illustration.

Case Study

Systems Being Verified

For generalization, the case is selected from a public source, demonstration of Matlab.

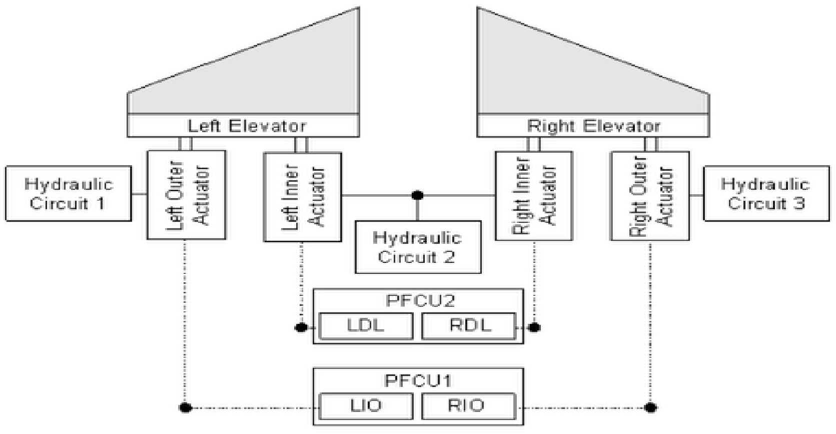


Fig 1. Schematic showing how the components of the elevator system are connected to one another

Aircraft elevator may be the appropriate system for our study. A typical aircraft has two elevators attached on the horizontal tails. And they are distributed on both side of the fuselage named left elevator and right elevator. There are number of redundant parts in the system to enhance safety of the aircraft. As the figure 1 shows the schematic of the components of elevator system are connected to another.

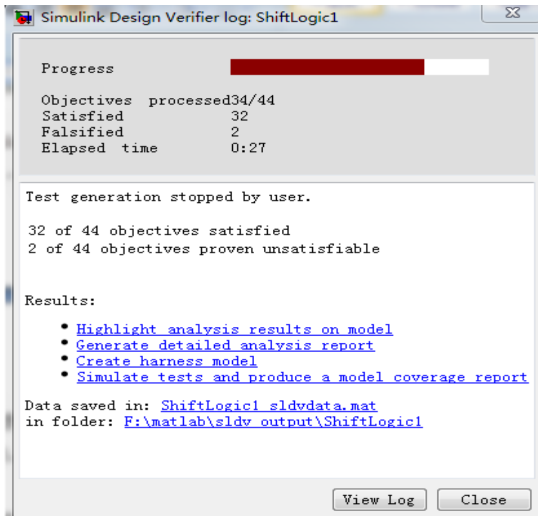
There are two independent hydraulic actuators per elevator, and three separate hydraulic circuits to drive the actuators. PFCU1 and PFCU2 are the two primary flight control units. PFCU1 is connected with the left outer actuator and right outer actuators. PFCU2 is connected with the left inner actuator and right inner actuators. Two control modules per actuator are used to regulate the full range control law and limited/reduced range control law.

For other detailed information, please refer to demonstration of Matlab.

Testing

This subsection will discuss the method of testing using Matlab verification and validation tools. Simulink Verification and Validation automates requirements tracing, modeling standards compliance checking, and test-harness generation. It can also provide modeling standards checks for the DO-178BC.

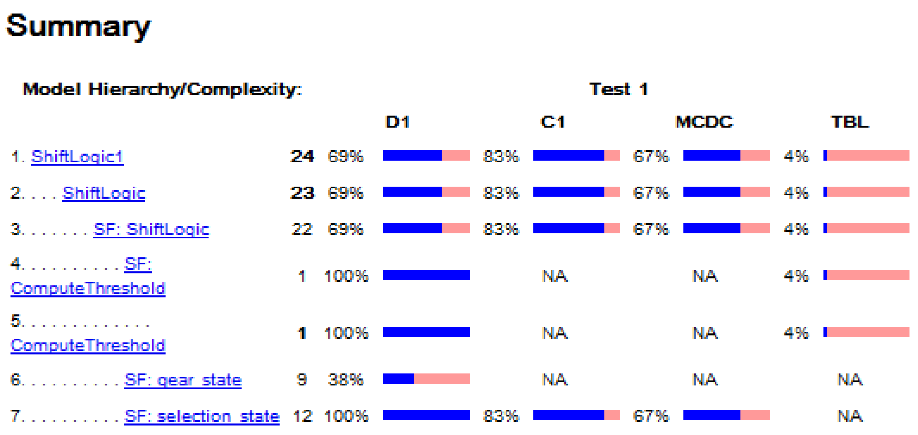
The point of this section is to introduce the methods to run the test automatically using verification and validation tools. After modeling the actuator software using Simulink we can create the test for the system shown in figure 2:


Simulink Design Verifier log: ShiftLogic1

```
Progress

Objectives  processed34/44
Satisfied             32
Falsified             2
Elapsed  time         0:27

Test generation stopped by user.

 32 of 44 objectives satisfied
 2 of 44 objectives proven unsatisfiable

Results:

    • Highlight analysis results on model
    • Generate detailed analysis report
    • Create harness model
    • Simulate tests and produce a model coverage report

Data saved in: ShiftLogic1_sldvdata.mat
in folder: F:\matlab\sldv_output\ShiftLogic1
```

Fig. 2. Creating test using Simulink Verification and Validation tools.

By click the item that needed, the verification tools can create the coverage report automatically, shown in figure 3.


Summary

Model Hierarchy/Complexity:		Test 1 D1	C1	MCDC	TBL
1. ShiftLogic1	24	69%	83%	67%	4%
2. . . . ShiftLogic	23	69%	83%	67%	4%
3. . . . . . . SF: ShiftLogic	22	69%	83%	67%	4%
4. . . . . SF: ComputeThreshold	1	100%	NA	NA	4%
5. . . . . ComputeThreshold	1	100%	NA	NA	4%
6. . . . . SF: gear_state	9	38%	NA	NA	NA
7. . . . . SF: selection_state	12	100%	83%	67%	NA

Fig. 3. Model coverage report

Formal verification

In this subsection, we will discuss the formal verification method. The tool used below is Processed Analysis Toolkit (PAT), which is developed by National University of Singapore [4]. PAT is a self-contained framework for to support composing, simulating and reasoning of concurrent, real-time systems and other possible domains. It comes with user friendly interfaces, featured model editor and animated simulator. Most importantly, PAT implements various model checking techniques [5] catering for different properties such as deadlock-freeness, divergence-freeness, reachability, LTL properties with fairness assumptions, refinement checking and probabilistic model checking.

The point of this section is to introduce the main path to model the actuator control software system using the language of PAT:

1) Translate the Stateflow of Matlab model to language used in PAT, the step has to done artificially;
2) After complete modeling the actuator using PAT, the next thing to do is run the PAT simulation and verification. PAT can verify and check the model with different aspects, such as logical deadlock, and the trace of status;
3) PAT can also check the property of model. We can assert a property using PAT language, run the verification, and PAT will check if the model satisfies the property. If not, the PAT will illustrate with a counterexample. Figure 4 show a sample of a counterexample.

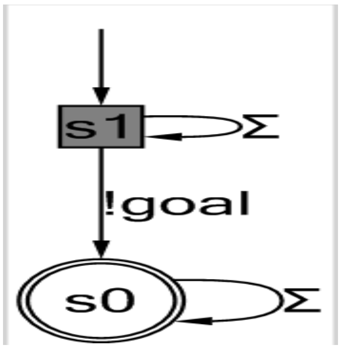


Fig. 4. The theory of counterexample

Summary

The primary motivation for this paper is to compare formal analysis and testing for verification of safety-critical systems. The formal analysis can reduce the cost and improve the effectiveness of safety verification. And it can provide with the detailed information of the software that need to be verified. However, in some case, the test is still needed, for example in system integrating.

There are still some tasks to be done. The first thing is to improve the correctness of the PAT model of the software. For now, we have to translate the mdl to PAT file manually, which is error-prone. The next step of the work includes developing a method to translate the mdl file to PAT automatically. We think the problem we met is general for similar works. From the case study, you could see the advantage and limitation of testing and formal method in verification of safety-critical systems.

References

[1] RTCA (Radio Technical Commission for Aeronautics) and EUROCAE (European Organisation for Civil Aviation Equipment). DO-178C/ED-12C: Software Considerations in Airborne Systems and Equipment Certification. (2011)

[2] B. Duncan, D. Hervé, H. Kelly and W. Virginie. Guidance for Using Formal Methods in a Certification Context, in ERTS 2010 – 19-21, Toulouse, May 2010

[3] C. Cyrille, K. Johannes and M. Yannick. Integrating Formal Program Verification with Testing, in: ERTS 2012, Toulouse, Feb. 2012

[4] PAT: Process Analysis Toolkit An Enhanced Simulator, Model Checker and Refinement Checker for Concurrent and Real-time Systems, available at http://www.comp.nus.edu.sg/~pat/

[5] P. Doron, P. Patrizio and S. Paola. Model Checking, Wiley Encyclopedia of Computer Science and Engineering, 2009.