AIRBORNE SOFTWARE VERIFICATION FRAMEWORK AIMED AT AIRWORTHINESS

Yumei Wu*, Bin Liu*
***Beihang University**

Keywords: *software airworthiness, software verification framework, software reliability and safety*

Abstract

Software verification framework aimed at airworthiness (SVFAA) during the development process is presented based on the software requirement of airworthiness. Activities and strategies included are illustrated in detail. To implement the SVFAA in the engineering practice is introduced. Setting up internal verification team and routines is also discussed, which is a very useful way to ensure the effectiveness of SVFAA and related work.

1 Introduction

The requirements of airworthiness for software can be abstracted from the analysis of DO-178: (1) the certification liaison process is added, during which conformity proof and software life cycle material should be presented; (2) development and verification tools should be certified; (3) the whole document system should be complete, the documents of PSAC, software verification plan, software verification procedures should be included in the document system; (4) the verification work during the whole software life cycle should be carried out. All these emphasize the conformity check and the verification work which unfortunately has only been partly done in our country nowadays.

The software development process based on the GJB (Chinese military standards) does emphasize the importance of testing, but does not emphasize the verification work of the development process, which is particularly and strongly required by DO-178B. This standard states the objectives to meet and the rules to follow for the software under development in order to have airworthiness certification together with the airborne systems or equipment. Based on the pre-research work and analysis, considering the present situation of military software development, the software verification framework is presented which is discussed and explained thoroughly in this paper.

2 Verification Framework

Verification is the act of reviewing, inspecting or testing, in order to establish and document that a product, service or system meets regulatory or technical standards [1]. Verification is not just testing. In fact testing is part of the verification process, analyses and reviews should be included as well. Reviews are qualitative and generally only performed once, whereas analyses are more detailed and reproducible. They are both used to confirm whether the outputs of different processes in development process satisfy the objectives.

The verification framework is composed of the activities (in this section) and strategies (in section 3) which try to help airborne software development institutes to set up the verification mechanism properly and carry out the verification work efficiently. So the specific verification techniques are not included in this paper.

2.1 Purpose of verification

The purpose of the software verification process is "to detect and report errors that may have been introduced during the software development process" [2]. It defines verification objective, rather than specific verification techniques, since the later may vary from one project to another or over time. The verification

work should be accomplished during the whole software lifecycle process.

The lifecycle processes specified in GJB 2786 and DO-178B is quite different in terms of the partition of the processes, but almost the same in essence which is possible for us to make the mapping work of the software life cycle process between these two standards (see Fig. 1) This is very important to carry on software engineering work within the GJB framework, and meet the requirements of DO-178B.

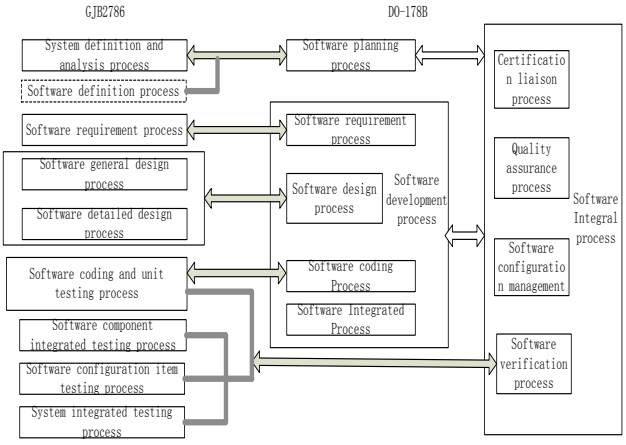

Fig. 1 The mapping work between GJB 2786 and DO-178B

The software development process based on DO-178B can be seen more clearly in Fig.2. In this paper we are talking about the verification work during the development process: the software requirements process, which produces the high-level requirements (HLR); the software design process, which produces the low-level requirements (LLR) and the software architecture through one or more refinements of the HLR; the software coding process, which produces the source code; the integration process, which produces executable object code and builds up the integrated system or equipment.

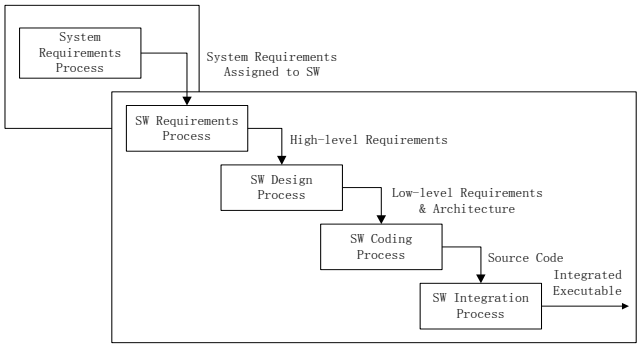

Fig. 2 Software development process

2.2 Activities in the verification process

The objectives of the verification process in the development process are illustrated in DO-178B. The appropriate verification activities in the development process are simply illustrated in Fig.3:

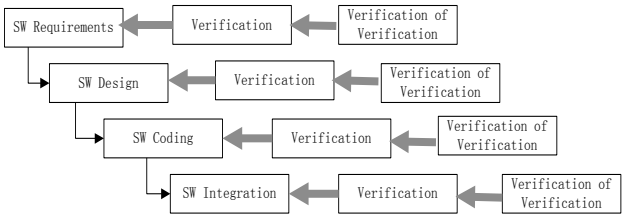

Fig. 3 Verification activities in the verification process

- Verification is used to confirm whether the outputs such as HLRs, LLRs, software architecture，the source code and object code of different processes in development process meet the objectives (see Table 1). Fig.3 shows that verification related activities should be carried out in software requirement process, design process, coding process and integration process.
- The verification process itself which is called verification of the verification activities should be verified as well (see Table 1) which include Test procedures, Test results, Requirement coverage, Structure coverage such as statement coverage, branch coverage, decision coverage, Data and control coupling.

Table 1: Content of the reviews and analyses

Objects	Objectives
High-Level Requirements	Compliance with the system requirements Accuracy and consistency Compatibility with target computer Conformance to standards Traceability; Algorithm aspects
Low-Level Requirements	Compliance with HLRs Accuracy and consistency Compatibility with target computer Verifiability Conformance to software design standards Traceability; Algorithm aspects
Software architecture	Compliance with HLRs Consistency Compatibility with target computer Verifiability Conformance to standards partitioning integrity
Source Code	Compliance with LLRs Compliance with the software architecture Verifiability

	Conformance to standards Traceability Accuracy and consistency
Outputs of the integration process	Incorrect hardware addresses Memory overlaps Missing software components

3 Verification Strategies

Strategies can be thought as solutions of meeting the verification objectives. The methods, the ways of review and analysis, even the tools used should be included in the strategies. The following parts cover the different aspects of the strategies based on the explanation of the verification objectives.

3.1 Verification of test procedures and test results

The following aspects of test procedures have to be reviewed for correctness:

- Test procedures for the high-level requirements.
- Low-level test procedures for compiler verification.
- System or high-level requirements based test procedures on the target.

The results of the above test procedures have to be reviewed and analyzed. Discrepancies have to be explained.

3.2 HLR coverage analysis

The objective of this activity is to verify that all the HLRs have been covered by test cases. All test cases are based on HLR and contain links to HLR that they verify. The analysis of these links can confirm the full coverage of the HLRs by the test cases. Or these test cases have to be complemented. All HLRs have to be fully covered by target testing. In many cases it is time and resource consuming particularly for the large scale software. Simulation testing is used to replace full coverage of HLRs in target testing. If the representativeness of the simulation is demonstrated, then the coverage has to be ensured by the union of test cases performed on the target and on the simulator.

3.3 LLR coverage analysis

The most disadvantage of human natural language is ambiguity which means that there will be different understandings for one simple statement. It leads to unintegrity, inconformity and even wrong understanding in software development processes. Thus formal language is used in modeling to solve this problem.

Model coverage analysis is a method of assessing how far the behavior of a model is explored. It is complementary to the traceability analysis and HLR coverage analysis. It helps verify that every element of the model is dynamically activated when the HLRs are exercised [4]. A primary objective is to detect unintended functions in the software design. Coverage criteria are used to analyze the dynamic behavior of the software design which includes control flow coverage criteria and data flow coverage criteria. More specific coverage criteria or rules are added based on different modeling methods. After acquisition and analysis of the model, adding test cases or providing explanation may be used to fix the model or HLRs. Thus by these fixes and model coverage, deficiencies will be revealed, such as shortcomings in HLR based test procedures, inadequacies in HLRs, dead parts or deactivated parts in the model.

3.4 Structural coverage of the source code

Statement coverage, branch coverage and path coverage are applied during the structural coverage verification of the source code. The structural coverage objective depends on the safety level of the software, for instance, the software of Level A requires MC/DC (modified condition/decision coverage).

Test cases ensuring MC/DC coverage of all the basic C blocks are developed. They are both exercised on the host and the target processors. Then one can assert that every required computational path through the generated code for the primitive computational block level has been exercised correctly. Several commercial tools for instance SCADE CVK and LDRA TBrun are used to do MC/DC for the software of Level A.

3.5 Data coupling and control coupling

The verification of data and control coupling depends on the hardware/software interface, software architecture, code structure, and source code language. The verification of data and control coupling involves a combination of: reviews and analysis of Software Architecture, reviews and analysis of source code and testing.

Certification authorities have observed a number of problems with the application of data coupling and control coupling analyses to airborne software [4]. The strategies below help to proactively address these issues:

- The data coupling and control coupling analyses in their plans should be addressed. In some cases, it may be distributed among several plans. However it is documented, it must provide complete and accurate rationale (i.e., it must be thorough).
- Data coupling and control coupling as part of their development/design effort (i.e., specify interface (I/O) requirements and dependencies between components) should be considered.
- The rationale for "analysis" and/or "testing" aspects of satisfying the objective should be substantiated. The objective may be satisfied as a static activity (e.g., looking at a link map or call tree), a dynamic activity (i.e., running tests), or a combination.
- That data and control coupling analysis are two different analyses (i.e., they should not be combined into one activity).If tools are used, the determination of whether they need to be qualified or not should be evaluated and justified.
- If selective linkers are used, their effect on data coupling and control coupling must be analyzed.
- The verification methods to satisfy the objectives of verification of verification process are summarized in Table 1.

Table 2: DO-178B A-7 objectives achievement

Objective	Verification Strategy
Test Procedures are correct.	Test procedure review
Test results are correct and discrepancies are explained.	Test results review
Test coverage of high-level requirements is achieved	HLR coverage by combination of simulation and target tests
Test coverage of low-level requirements is achieved	Model test coverage
Test coverage of modified condition/decision is achieved.	MC/DC
Test coverage of decision coverage is achieved.	Decision Coverage
Test coverage of statement coverage is achieved.	Statement Coverage
Test coverage of data coupling and control coupling is achieved.	Semantic check of the model and manual verification of integration with manual code

3.6 Tools

In order to improving verification efficiency process automation and process elimination are mainly focused on:

The use of software tools to automate activities of the software life cycle processes can help satisfy system safety objectives as long as they can enforce conformance with software development standards and use automatic checks.

The verification of the development process can be reduced or eliminated on the premise of that the tool provides confidence at least equivalent to that of the process. Thus the verification process in the development cycle is changed to the tool qualification process.

4 Engineering practice

We are now making application plans to add verification work in one of the military aircraft software projects under development based on the SVFAA, which helps work out "what to do", "when to do", and "how to do".

The document system is set up including the templates of software verification plan, software verification procedures provided.

Checklists are made for efficient review to check if the outputs meet the objectives such as compliance with the up level requirement, accuracy and consistency, compatibility, conformance to standards.

Strategies presented in the SVFAA are assigned to different processes. And we have also made guidelines which help explain the strategies in detail and instruct the procedures by defining the transit rules.

Several considerations about management, safety and reliability related requirements are also added in implementation of the SVFAA.

(1) Management consideration

People who are engaged in software verification work are divided into three levels in organization (see Fig.4). It is completely separated from the development line, which can guarantee the independence of the software verification work.

Software testing staff are responsible of doing verification work according to the verification plan within the SVFAA.A unified internal testing and verification process should be set up by software verification team. Guidelines and schemes should be made accordingly for internal testing staff to accomplish the verification work within the SVFAA. The software verification team is also in charge of making the verification plan, collecting and summarizing the results of the verification activities from the software testing staff. Review and analysis of the results should be carried out as well.

A reporting mechanism should be run efficiently. All the results are presented to the software chief engineer to assess effective implementation of the verification plan and activities, make determination on acceptable deviations from plans and standards found during the review.

Fig. 4 Software verification management

This could be considered as "who will do".

(2) Safety and reliability considerations

For the safety critical modules and functions strategies as follows are used to verify the requirements of reliability and safety.

- During the software requirement and design process software hazard analysis should be applied to analyze the hazards which could be caused by the defects of the software input, running and related supporting environment, and function design, and also to verify the related measures have been taken to prevent the hazards from happening. FMECA, FTA, information flow analysis, dynamic simulation can be used during this process.
- In the software coding process, code safety analysis is strongly suggested to find the defects which cannot be detected in the unit testing and common code review. Code safety analysis is especially efficient to find software defects related with the hardware constraints, and caused by the complicated conditions which in most cases happen to the embedded real-time software.
- During the software integration process, apart from the integration testing and system testing reliability testing should be carried out based on the software reliability verification plan and operational profile to meet the reliability index requirement for the software if there is any.
- The software reliability and safety evaluation should be made after the related work during the development process, of which the verification results of the reliability and safety are included.

5 CONCLUSIONS

It is the first time that the software verification work aimed at airworthiness is applied in the development of one of the military aircraft software projects. It will definitely improve the quality especially the reliability and safety of the airborne software. The experience accumulated during the execution of the schemes and guides based on the verification framework will be very helpful to strengthen the software engineering infrastructure used nowadays.

The effect of doing verification work during the software life cycle process will be investigated and changes of the SVFAA will be made to match the requirements and meet the objectives. Also the related verification techniques will be studied further by more researchers in more scale.

REFERENCES

[1] Space Engineering Verification [S]. ECSS-E-10-02A, 1998.
[2] Software Considerations in Airborne Systems and Equipment Certification [S]. RTCA/EUROCAE, 1992.
[3] Defense System Software Development [S].COSTIND, 1996.
[4] Clarification of Structural Coverage Analyses of Data Coupling and Control Coupling [C], CAST-19, 2004.

Copyright Statement

Contact Author Email Address

wuyumei@buaa.edu.cn