A Safety Analysis Method of Airborne Software Based on ARP4761

Rui Sun¹, Deming Zhong¹, Weigang Li², Minyan Lu¹, Yuxin Ding¹, Zhi Xu¹, Haoyuan Gong¹ and Yun Zha¹*,

¹Beihang University
²China AECC AERO Engine Control System Institute, China
Email: eenh343r@gmail.com
*Corresponding Author Email: zy1914102@buaa.edu.cn

Abstract. Most of the functions of civil aircraft are achieved through software at present. Therefore, the safety of airborne software has become an important part of safety analysis. However, to achieve complex functions, the onboard software becomes very complicated, and it is usually difficult to guarantee safety. ARP4761 (Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment) [1] is an important safety assessment guideline for civil aircraft, but it believes that software development can satisfy safety as long as the software development process complies with the software development standards, instead of incorporating the software into the analysis process. To improve airborne software safety, this paper proposes a safety analysis method based on the ARP4761 process which is tailored to adapt to the software. In the software requirements and design, the method uses FHA(Functional Hazard Assessment) to analyze the hazards of the software and define the hazard levels, then use FTA(Fault Tree Analysis) to build a software fault tree, and carry out CCA(Common Cause Analysis) analysis based on the fault tree; after the software development is completed, the method uses FMEA(Failure Modes and Effects Analysis) and FMES(Failure Modes and Effects Summary) to summarize the failures of the software and return these failures to the fault tree to determine whether the software can meet the defined requirements. At the end of the article, an anti-icing software is used as an example to explain the process and effectiveness of the method.

1. Introduction
Most of the functions of civil aircraft are achieved through software at present. Therefore, the safety of airborne software has become an important part of safety analysis. However, to achieve complex functions, the onboard software becomes very complicated, and it is usually difficult to guarantee safety. ARP4761 (Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment) [1] is an important safety assessment guideline for civil aircraft, but it believes that software development can satisfy safety as long as the software development process complies with the software development standards (DO-178C[2]), instead of incorporating the software into the analysis process. Serious safety problems may be caused if an engineer only analyzes the safety of aircraft based on ARP4761 and ignores the impact of software. For example, one of the causes of the Boeing 737MAX-8 accident[3] is the incorrect software logic in the thrust enhancement system.
To solve the safety problem of software, researchers have proposed many analysis methods, such as software FTA (Fault Tree Analysis) [4], software FMEA (Failure Modes and Effects Analysis) [5], STPA (Systems Theory Process Analysis) [6], formal method [7], and the integrated method between these methods, such as the integrated method of STPA and FMEA [8]. However, these methods are not used in the ARP4761, so these methods are difficult to incorporate into the existing safety analysis results, which eventually leads to insufficient software safety analysis results.

To solve this problem, this paper proposes a safety analysis method based on the process of ARP4761 which is tailored to adapt to the software. In this way, the safety analysis method proposed in this paper can be seamlessly connected with ARP4761, and the process and results of different security analysis methods are not compatible with the existing civil aircraft safety analysis system. This paper is divided into four chapters, the first chapter is the introduction, the second chapter introduces the steps of the method, the third chapter is an example of the method applied to an anti-icing software, and the fourth chapter is the conclusion of the method.

2. The Software Safety Analysis Method
Based on the software development process, we divide the process of software safety analysis into three parts: requirements and design phase, implementation phase, and verification phase. Therefore, we tailor the process of ARP4761 and divide it into three stages according to the characteristics of the software, as shown in Figure 1.

2.1. Software Requirements and Design
The purpose of safety analysis in software requirements and design is to check the structure of software, determine its function and corresponding hazard attributes, and judge its hazards impact on the system. This part consists of three steps: software FHA, software CCA, and software FTA. Among them, software FHA is used in the requirement, while CCA and FTA are used in the design.

Software FHA is mainly used to identify the failure status of software-intensive items, determine the severity, and classify them according to the impact of failure states. According to the classification of failure states, the corresponding safety requirements of failure states can be determined. It is worth noting that since the hazard of software is determined from the system, FHA of software-intensive items rather than the software itself must be carried out here.

Software FTA is a top-down analysis method that is used to determine the cause of software function failure. It takes important failure states in FHA as the top event, which is and the bottom event of the FTA is the failures of hardware and software components. The output of software FTA is hazards in the software.

Software CCA is used to supplement and improve the software FTA and its results. The premise of FTA analysis is that the system is strict, so CCA is needed to ensure the independence of FTA analysis. The software CCA which is based on the input of software requirement, design, and software failure tree, is used to verify the independence of "and gate" events in the software FTA, qualitatively analyze the potential common cause hazard status of software-intensive items, and output the software requirements and design hazards related to the common cause, to ensure that the software related common cause hazards are eliminated or controlled.

2.2. Implementation
In the implementation part, engineers implement the corresponding software according to the safety requirements of software obtained from software requirement identification and hazard analysis. The implementation does not require a safety analysis, it is only a process of software development.

It should be noted that some software safety requirements will eventually be implemented in the hardware (such as hardware response time). Therefore, some hardware related implementation will also appear here.
2.3. Software Verification

The purpose of hazard analysis in the software verification part is to analyze and verify whether the safety requirements determined in the requirements and design part are met according to the results of software and hardware item design and implementation.

This part consists of three steps: software FMEA and FMES, software fault tree update, and software common cause analysis update. As some hardware failures will affect software failures, these failures need to be considered by software. Therefore, software FMES may need to include some hardware FMEA results, but hardware FMEA is not included in the process.

Software FMEA is referring to the failure mode checklist of software code, analyze the failure mode of software code designed and implemented. According to the actual work, the failure mode of FMEA can be a software component level or software unit level. FMES is the summary of FMEA, its purpose is to determine the impact of code failure on software.

Software fault tree update is an update process that based on the implemented software, the structure and data of the fault tree constructed in the software requirements identification stage are updated through the software FMES result, to verify whether the safety of the implemented software meets the initial requirements.
Software CCA update is an update process based on the updated software fault tree. The purpose of the step is to verify whether the designed software still meets the requirements of independence. If the previous software fault tree analysis and independence requirements are not updated, this step is not required.

3. Case Study
This method is applied to aircraft anti-icing software. An anti-icing software is a part of the anti-icing system, which includes manual anti-icing, automatic anti-icing, failure alarm of anti-icing, and power-on self-check of anti-icing, the process is shown in Fig. 2.

![Figure 2. The Process of Anti-ice](image)

3.1. Requirement and Design of Anti-ice Software

3.1.1. FHA of the software The purpose of FHA is to identify the function of the anti-icing control system, functional hazard mode, and related hazard attributes. According to the influence of failure state, the effect is determined and classified, and the FHA result is given, which includes three parts: function failure state analysis, work stage analysis, and hazard classification, then determine the failure rate requirements of failure condition based on the result of FHA.

| Function          | Failure Condition | Phase | Effect of Failure Condition on Aircraft/Crew | Classification | Verification |
|-------------------|-------------------|-------|---------------------------------------------|----------------|--------------|
| Anti-ice software | Failure of anti-ice control function | F0,F1,F2,F3,F4,F5 | Personnel injury; flight mission interruption; serious engine damage | II | FTA |
|                   |                   | F0    | Harmless to personnel; no effect on the mission; no damage on engine | IV             |              |
|                   | Can’t turn off of anti-ice function | F0,F1,F2,F3,F4,F5 | Harmless to personnel; no effect on the mission; no damage on engine | IV             |              |
|                   | false opening of the anti-ice function | F0,F1,F2,F3,F4,F5 | Harmless to personnel; no effect on the mission; no damage on engine | IV             |              |

Firstly, the failure status of the anti-icing software is analyzed, including the failure of the anti-icing control system, unturn off of anti-ice function, and the false opening of the anti-ice function.
Secondly, the corresponding working stages of fault states are determined, including F0: ground operation; F1: taxiing and taking off; F2: climbing; F3: leveling flight; F4: landing; F5: landing and taxiing.

Finally, the hazard of anti-icing software failure in each working stage is analyzed by matching the function failure status and working stage, and the FHA analysis result is given, as shown in Table 1.

According to the failure conditions and corresponding safety classification of the automatic anti-icing control, the safety requirements can be derived (only considering the failure condition with the classification as above III) as the failure rate of anti-icing (F1, F2, F3, F5, F6) shall be less than 5e-7 per flight; (in the safety requirements assigned below, the average time of each flight mission is 5 hours)

3.1.2. FTA of the software Software FTA analysis is a downward decomposition of safety requirements based on the Failure Condition obtained from the FHA. The safety requirements of the software are derived to the design. The agreed levels of events at the bottom of the Fault Tree are software item design (SI) and a part of hardware item design (HI). The fault tree is shown in Figure 3.

![Figure 3. The Fault Tree of Anti-ice Software](image-url)
3.1.3. **CCA of the software** The functions of anti-icing control software must be independent to ensure that there is no common mode fault due to development error. By analyzing the “and gate events” of the fault tree in Fig. 3, it can be concluded that the independent events of software CCA include: 1) The channel A and B of anti-icing software shall be independent of each other to ensure that the failure of anti-icing function in a single channel will not cause the failure of the anti-icer; 2) The fault and collaborative control of solenoid valves in each channel of anti-icing software need to be independent to ensure that single failure will not cause engine anti-icing function failure. The result of CCA is shown in Table 2.

| Failure condition | Examples of common mode source | Examples of common mode failures |
|-------------------|--------------------------------|----------------------------------|
| Both channel A and B failures software fails at the same time | Share the same CPU or memory address | CPU failures or memory leak |
| Fault control and collaborative control fail at the same time | Same input variables | an error in the input variables |

### 3.2. Implementation of Anti-ice Software

The implementation of the software is a part of software development but not a part of the safety analysis method. However, the analysis results of software FHA, FTA, and CCA should be included in the code to ensure the safety requirements of the anti-ice software.

### 3.3. Verification of Anti-ice Software

#### 3.3.1. **FMEA and FMES of the software**

According to the software test results, we carried out software FMEA. The failure modes used the software failure mode library summarized from different standards and experiences[9], then we summarized the result of FEMA through software FMES.

In order to adapt to ARP4761, the failure rate of the software is 0 or 1. If a software item passes the test, its failure rate is 0, otherwise it is 1. Since there are many analysis contents of FMEA, we do not purpose the result in this paper. Only the results in which the failure rate is 1 of software FMES are given, as shown in Table 3.

| Software item code | Failure mode of code | Failure rate | Failure effect |
|--------------------|----------------------|--------------|----------------|
| Anti-ice logic     | Automatic anti-ice has no output | 1 | Automatic anti-ice failure(due to design it will stay opening) |
|                    | Cannot exit anti-ice control logic | 0 | No safety effect |
| Collaborative control | Still entering this channel in the condition of channel failure | 1 | Automatic anti-ice failure |
| Fault control      | Fault control is abnormal | 1 | Normal channel is misdiagnosed as an error |

### Table 3. FMES of the anti-ice software
3.3.2. **Software CCA update** we obtain the possible common mode failure results of the software in section 3.2. Through the analysis results of FMEA, it can be found that these problems do not exist in the designed software, so the CCA analysis of the software does not need to be updated.

3.3.3. **Software FTA update** According to the analysis results of FME, we found that some bottom events in the software fault tree appeared in the software. Therefore, we need to update the software fault tree and recalculate the rate of the top event. The result is shown in Figure 4.

By comparing the fault tree before and after the update, it can be seen that the actual rate of "anti-icing control function failure" is 8.16e-7 per flight, which is higher than the safety requirements of 5.0e-7 determined by FHA. The failure rate does not meet the safety requirements, therefore, collaborative control software and fault control software needs to be modified to meet the failure rate requirements.

![Figure 4. The Updated Fault Tree of Anti-ice Software](image)

4. **Conclusion**
This paper presents a software safety analysis method based on ARP4761. The method retains the basic characteristics of ARP4761, which can easily integrate the analysis results into the process of
ARP4761, maintain the consistency of the safety analysis process, integrate the software safety analysis process into the analysis process of the whole system, and enable the analysts to quickly carry out the software safety analysis.

We apply the method to an anti-icing software, the results show that there are defects in the anti-icing software, and the defects will lead to the failure rate of the whole anti-icing control system higher than the requirements, so it must be modified. The application shows the effectiveness of the proposed method, but the method still has some limitations. It is mainly reflected in two aspects: 1. The software analysis is still carried out according to the system process, not the software operation profile; 2. The failure mode of software FMEA is closely related to the analysis quality.

In the following, our further research plans include two points. One is to consider the scenario of software failure through an operation profile; the other is to extend the failure mode of software FMEA.

5. Reference
[1] ARP 4761. Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment[S]. U.S.: SAE International, 1996.
[2] RTCA/DO-178C: Software Considerations in Airborne Systems and Equipment Certification[S]. RTCA, 2011.
[3] Aircraft Accident Investigation Report of PT. Lion Mentari Airlines Boeing 737-8 (MAX); PK-LQP. http://knkt.dephub.go.id/knkt/ntsc_home/ntsc.htm, Accessed 08/21/2020
[4] M. Takahashi, R. Kosaka, R. Nanba, Y. Anang and Y. Watanabe, "A study of methodology for securing control software based FMEA-FTA coordination," 2016 IEEE/SICE International Symposium on System Integration (SII), Sapporo, 2016, pp. 144-149, doi: 10.1109/SII.2016.7843989.
[5] Andrea Bondavalli, Francesco Brancati, Certifications of Critical Systems - The CECRIS Experience, Chapter 9, Composable Framework Support for Software-FMEA through Model Execution., 2017
[6] Asim Abdulkhaleq, Stefan Wagner, A controlled experiment for the empirical evaluation of safety analysis techniques for safety-critical software, Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, 2016.
[7] Asim Abdulkhaleq, A Comprehensive Safety Engineering Approach for Software-intensive Systems Based on STPA[A], 3rd European STAMP Workshop, 2015
[8] Sulaman, S., Beer, A., Felderer, M. et al. Comparison of the FMEA and STPA safety analysis methods–a case study. Software Qual J 27, 349–387 (2019).
[9] Yuxin Ding, Deming, Zhong, at el. An Airborne Software FMEA Application Method Based on ARP4761. 2019 2nd International Conference on Management Engineering, Software Engineering and Service Sciences