A Framework for Airborne Aviation Software Safety Requirements Analysis

ZHANG Yifan BAO Xiaohong LI Zhen
Dept. of System Engineering of Engineering Technology
Beihang University, Beijing, P.R.China, 100191
ivanbuaa@hotmail.com

Abstract: This paper presents a method to carry out the work of safety requirements for software in safety-critical systems, particularly in the aeronautical engineering domain. This method is based on the risk control idea. We make series iterative steps to develop full considered, risk rank classified software safety requirements, in order to support software developing work and make a guide for resource allocation and activity tailoring. In this paper, we propose an effective and operable framework which combines with the existing software engineering process well, software organizations can generate and classify software safety requirements to guide their following development process.

Keywords: Requirements, Software safety, Framework

1 INTRODUCTION

With the complexity of airborne aviation system growing, close coupling of hardware and software, software's size, complexity and its importance throughout the system increasing, more and more functions are performed by software instead of hardware. According to Hecht and Buettner[1], in the period of 1998 to 2000, nearly half of all observed spacecraft anomalies were related to software. Software reliability and safety issues have become more and more prominent. The characteristics of high safety and high mission reliability in Aviation areas determine that the failures of core control software may result in serious equipment damage or damage to property, and even threatening the lives of officers. Most aircraft companies develop onboard systems software for civilian or military aircraft based on the guidelines RTCA DO-178B. This, apart from ensuring the reliability of the software, also Require us to particular emphasis software safety issues[2].

However, RTCA DO-178B is Goal-oriented and emphases the processes, it doesn't supply specific implementation steps and technical methods. Besides, the works described in RTCA DO-178B are carried out under the premise of achieving complete system safety requirements. Especially in RTCA DO-178B section 1.1 and 2.1.1, it puts forward the demand for the safety requirements clearly. So we need to establish specific programs to satisfy the demands of software safety.

2 FRAMEWORK FOR AIRBORNE AVIATION SOFTWARE SAFETY REQUIREMENTS ANALYSIS

The objectives of Software reliability is to address how to reduce software failure, but the focus of the Software Safety work is to avoid or reduce the hazards associated with the software occur[3]. From the difference between software reliability and safety, we can see that software safety only concerns failures that may lead hazards occur, the associated design flaws and external input conditions. So compared with software reliability, the work on software safety has its own characteristics. For software safety, we should be emphasis on software safety requirements played an important role on developing safety software.

Many researchers affirm that the generation of software requirements is the major source of errors in system development. Most problems introduced into software can be traced directly to requirements flaws. Also, Leveson stated that in the space project domain, the vast majority of software-related accidents were related to flawed requirements and misunderstanding about what the software should do[4][5]. So, Software safety requirements work is an important part of the whole software safety work.

The primary purpose of our research is to propose an effective and operable framework which combines with the existing software engineering process well, software organizations can generate and classify software safety requirements to guide their following development process.

In order to connection with the existing software engineering processes closely, this framework is described according to the various stages of software developing processes, and it mainly includes the following five parts: System Risk Analysis; Identify the Basic Software Safety Risk; Generate the Basic Software Safety Requirements; Achieve Complete Risk of Software Safety Requirements; Result Output. Figure 1 illustrates the framework; the five parts in the framework and detailed stages in each part are explained as:

1. System Risk Analysis

The work of this part is mainly focus two core activities: Preliminary Hazard Analysis (PHA) and Functional Hazard Analysis (FHA). PHA is performed early in program to identify and prioritize hazards, it requires the knowledge of system design, architecture, functions, and so on. We can find Description of this method in MIL-STD 882. FHA is a systematic comprehensive top-down examination of functions within to identify and classify failure conditions of those functions according to their severity and assigning each failure condition probability requirements and applicable qualitative design requirements.

Due to the special nature of software, software itself is not dangerous, so we need to discuss software safety in the system environment[6]. Software safety requirements are comes from System safety requirements, only through a risk analysis of complex system layer by layer, identify risks that the system faced by, and to identify the degree that software associated with interrelated hazards, can we in-depth analysis to identify software safety requirements and the critical level of software safety requirements.

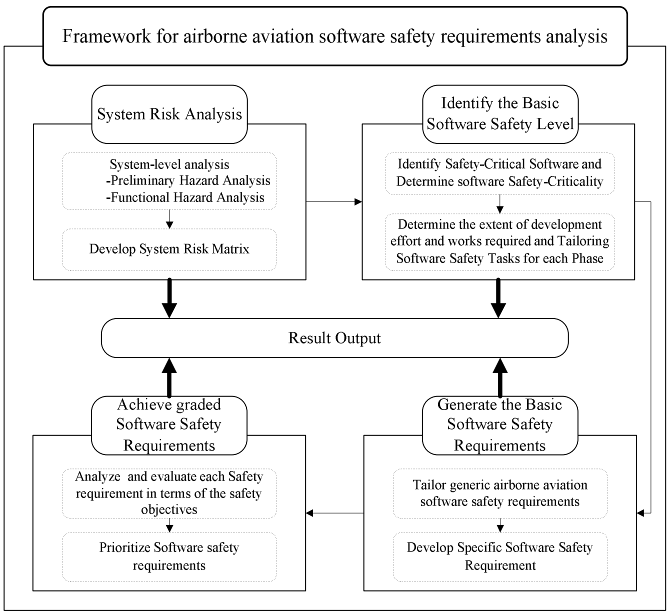

Framework for airborne aviation software safety requirements analysis
System Risk Analysis
System-level analysis
-Preliminary Hazard Analysis
-Functional Hazard Analysis
Develop System Risk Matrix
Identify the Basic Software Safety Level
Identify Safety-Critical Software and Determine software Safety-Criticality
Determine the extent of development effort and works required and Tailoring Software Safety Tasks for each Phase
Result Output
Achieve graded Software Safety Requirements
Analyze and evaluate each Safety requirement in terms of the safety objectives
Prioritize Software safety requirements
Generate the Basic Software Safety Requirements
Tailor generic airborne aviation software safety requirements
Develop Specific Software Safety Requirement

Figure 1 A framework for airborne aviation software safety requirements analysis

In the stage of System Risk Analysis, we consider the requests of ARP4761[7] and use FHA to help us to understand hazards the system will accrue and consequences which are made by function failures. The hazards must somehow be prioritized. This prioritization leads to the concept of risk.

Risk consists of two parts: Hazard Severity and Probability of Occurrence. According to ARP4761, Combining these two concepts (severity and Probability) leads to a single risk Level. This allows hazards to be prioritized and risks to be managed.

See Table 1 below. Highly severe hazards require a rigorous development and analysis environment to make its probability of occurrence low enough. While minor hazards require little or no extra attention, beyond the good engineering, programming, and assurance practices used by the project team.

Table 1 System Risk Matrix

Probability (Per flight hour)		Severity	Level
1.0~1.0E-3	Frequent	Minor	D
1.0E-3~1.0E-5	Reasonably Probable		
1.0E-5~1.0E-7	Remote	Major	C
1.0E-7~1.0E-9	Extremely Remote	Hazardous	B
<1.0E-9	Extremely Improbable	Catastrophic	A

2. Identify the Basic Software Safety Level

The mainly task of this part is to assign system risk to software, identify relevant safety-critical software risk and determine the level of required software safety effort according to the software Level. And we also tailor the works that are carried out during requirement stage, according to the extent of software safety effort. We can carry out by the following steps:

1) Identify Safety-Critical Software

Safety-Critical Software is the software that can cause a hazard or control a hazard directly; it also includes all the related software that can have an impact on the Safety-Critical Software. We can base on the characters of general software and consider the differences of airborne aviation software to identify.

In summary, according to NASA Software Safety Guidebook[8], software is safety-critical if it performs any of the following:

（1）Controls hazardous or safety-critical hardware or software.
（2）Monitors safety-critical hardware or software as part of a hazard control.
（3）Provides information upon which a safety-related decision is made.
（4）Performs analysis that impacts automatic or manual hazardous operations.
（5）Verifies hardware or software hazard controls.
（6）Can prevent safety-critical hardware or software from functioning properly.

2) Determine software Safety-Criticality

We can measure against the degree of safety critical software through: Degree of Control, Hazard related software, Complexity, Timing criticality, and so on.

Here, we improved the existing Degree of Control, make it more detailed and represent the characters of airborne aviation software. Finally the Degree of Control is divided into four grades as I, II, III, and IV.

We can conduct a preliminary classification of the safety-critical software that is analysis by the degree of control, and then consider the complexity of the software itself and other factors, classify again to get the software risk matrix, so that finalize identify software safety-Criticality.

Software risk Matrix is shown as follows:

Table 2 Software Risk Matrix

Software Category	Hazard Severity Level			
	Catastrophic	Hazardous	Major	Minor
I	1	1	3	4
II	1	2	4	5
III	2	3	5	5
IV	3	4	5	5

In this table, number1,2,3,4,5 respectively correspond to five software levels A,B,C,D,E which defined in RTCA DO-178B.

3) Determine the extent of development effort and works required

The extent of Software safety effort represents different requirements for resources, techniques and methods that are taken in the development of safety-critical software. According to statistics, the expense that we use to develop safety-critical software is about 100 times than that of common software. In order to effectively "optimize safety and reduce risks within the constraints of operational effectiveness, time, and cost throughout all phases of the system life cycle"[9], this requires us to determine the extent of safety effort of different types of software based on different software Level.

Required Software Safety Effort is shown in table 3

Table 3 Required Software Safety Effort

Software Level	Software Safety Effort
A	Full
B	Moderate
C	Moderate
D	Minimum
E	None

In different stages of software development, different extent of effort corresponds to different requirements of the work to tailor. We can refer to RTCA DO-178B's objectives, combined with the Specific methods listed in NASA Software Safety Guidebook to guide our tailoring work.

4) Tailoring Software Safety Tasks for each Phase

Once the scope of the software safety effort has been determined, it is time to tailor it to a given project or program. The safety activities should be sufficient to match the software development effort and yet ensure that the overall system will be safe.

The following software activities, such as development, analysis, inspections, reviews, verification and validation activities can be tailored, according to required software safety effort.

3. Generate the Basic Software Safety Requirements

This part we achieve complete software safety requirements through the tailor of generic software safety requirements, Fault and Failure Tolerance Considerations, Hazardous Commands Considerations, Timing, Sizing and Throughput Considerations, and so on.

1) Tailor Generic Software Safety Requirement

Similar processors, platforms, and/or software can suffer from similar or identical problems. Generic software safety requirements are derived from sets of requirements and best practices used in different programs and environments to solve common software safety problems. Generic software safety requirements capture these lessons learned and provide a valuable resource for developers. Generic requirements prevent duplication of effort by taking advantage of existing proven techniques and lessons learned rather than reinventing techniques or repeating mistakes.

In this regard, we work according to the status of researches at home and abroad[10][11][12][13], combining with the characteristics of airborne aviation software safety requirements and many years of accumulated experience[14][15][16][17], develop a list of general airborne aviation software safety requirements.

An example of generic airborne aviation software safety requirements is shown as table 4

Table 4 generic airborne aviation software safety requirements

CLASSIFICATION	SAFETY REQUIREMENT
System Design	The system shall have at least one safe state identified for each logistic and operational phase
	The software shall return hardware subsystems terms under the control of software to a designed safe state when unsafe conditions are detected.
	……

Especially in the aspect of interface, we consider the particularity of airborne aviation software. In designing computer-based systems, special attention must be give to software supporting interfaces such as human interface, input/output interface, and so on. The design of interfaces is not only critical for effective system performance, but it is also affecting the level of software safety[18]. Here, we develop a list of general airborne aviation software interface design requirements to help us complete generic software safety requirements.

An example of generic airborne aviation interface design requirements is shown as table 5

Table 5 generic airborne aviation interface design requirements

CLASSIFICATION	SAFETY INTERFACE DESIGN REQUIREMENT
Safety-critical alerts	Alerts shall be designed such that routine alerts are readily distinguished from safety-critical alerts.
	……

2) Develop Specific Software Safety requirements

Specific software safety requirements are system-unique functional capabilities or constraints that are identified in the following three ways. For complete identification of all software requirements, we should make a full analysis of the software and its system environment.

In order to achieve this goal, we can consider the following three methods:

（1）From the system-level analysis.

（2）Through top-down analysis of system design requirements and specifications.

（3）Through bottom-up analysis of design data.

To make an effective analysis requires us to consider various factors. In NASA Software Safety Guidebook we can find some specific guidance.

4. Achieve graded Software Safety Requirements

The main objective of this part is to connection the requirements to system risk by continuing in-depth analysis; determine the grade of software safety requirements and classify software safety requirements.

In the software developing process, not all "safety-critical" requirements are created equal. We analyze the hazards of the software/hardware system and identify those that could present catastrophic or critical hazards, evaluates each program requirement in terms of the safety objectives derived for the software component, use the concept of risk to prioritize which requirements are more critical than others.

With these, we can rational distribute resources of software development, to achieve the unity of cost and effectiveness. After achieved graded software safety requirements, we ensure that every safety requirement to be adequately tested by making the entire process tracing.

5. Result Output

With the consideration of military airborne aviation software developing processes, the outputs of this framework are documented, referred to GJB438A Defense system software development documentation. The outputs are mainly included in Software requirements specification, Interface requirements specification, and so on.

3 CASE STUDY

This section illustrates how we have applied our framework in an actual project generating and classifying software safety requirements to guide operators' following development process.

Engine digital electronic control system is an important part of aircraft engine system. It should be able to meet the needs of all of the engine control functions to ensure the engine in the flight envelope does not appear over-temperature, over-running, over-pressure phenomena. In this project Engine digital electronic control system is integrated by different subsystems. The electronic controller assumes digital operation and logical judgment, its work flow is shown below.

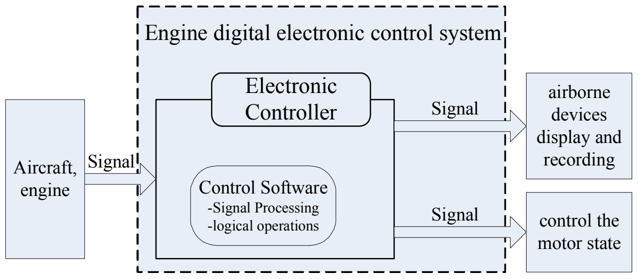


Figure 2 Electronic controller

To carry out system risk analysis, first we develop a preliminary hazard list with the help of system designers. This list is the initial set of hazards associated with the system under development. And it will mature during the following analysis process.

Table 6 Preliminary hazard list (part of)

Hazard Status	Level
engine surge	
engine over-temperature	
……	

According to System Design document, we sum up the system functions and then make a Functional Hazard Analysis for every sub function to identify safety critical functions.

Table 7 Function list (part of)

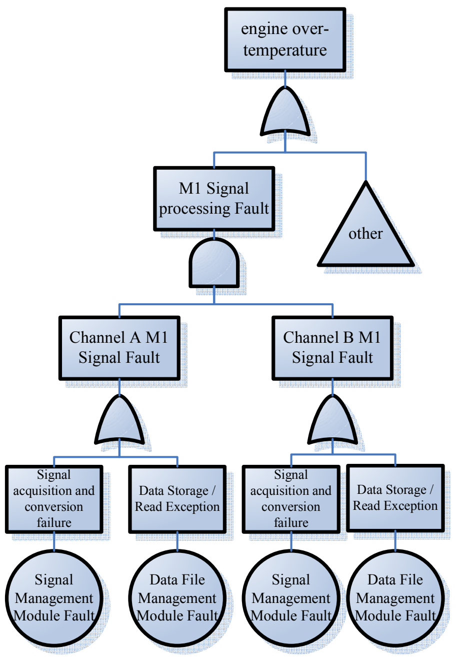
Top-level function	Expand level 1	Expand level 2	Function Number
Engine Fuel Flow Control	Flow control in starting process	Fuel supply control in ignition process	001
		Fuel supply control for accelerate in starting process	002
	……	……	……

Table 8 Functional Hazard Analysis (part of)

Function Number	Function	Failure Description	Working condition	Impact on the engine	Level	Measures
001	Fuel supply control in ignition process	Too much fuel supplied	ground	Start failure or deflagration	D	Targeted Design
			flight	Air starter unsuccessful	A	Further analysis
		Too less fuel supplied	ground	Start failure	D	Targeted Design
			flight	Air starter unsuccessful	A	Further analysis
……	……	……	……	……	……	……

Through Functional Hazard Analysis（FHA） above, we get impact degree of every function failure and propose improvement measures for each failure. According to these analysis results and other documents, we have a better understanding of this system. So we sum up eight top level hazards based on Preliminary hazard list. Then in order to identify software-related hazards causes, we use FTA as a Top-down analysis.

Fault Tree Analysis (FTA) is a top-down analysis technique that is used to identify the contributing elements (errors / faults / failures) that could precipitate the system level hazards identified[19, 20]. It is a feed-back technique in that one starts with the system level hazards and attempts to work backward by identifying all possible causes of the hazards[21]. For instance, for engine over-temperature:



engine over-temperature
M1 Signal processing Fault
other
Channel A M1 Signal Fault
Channel B M1 Signal Fault
Signal acquisition and conversion failure
Data Storage / Read Exception
Signal acquisition and conversion failure
Data Storage / Read Exception
Signal Management Module Fault
Data File Management Module Fault
Signal Management Module Fault
Data File Management Module Fault

Figure 3 FTA for engine over-temperature (part of)

After system risk analysis, we get an initial understanding of system safety state. And based on the second stage of work above, Reference degree of Control we determine the control software safety level as A.

When the safety level of control software is determined, it is easy for us to determine the extent of development effort and works required, and after the all steps above, we finally get basic graded control software safety requirements, effective improve the following software developing processes.

4 CONCLUSIONS

The framework supplies a solution that can meet the needs of generate, manage, trace and classify software safety requirements. The software organizations can benefit by applying the proposed framework to their own software development process to satisfy the need of DO-178B. The software safety for the whole system can be improved through the framework.

Now we are applying this framework in one project to gather experience, in the following work, we will improve this framework and make it integrate with the software engineering better.

REFERENCES

[1] M.Hecht,D.Buettner “Software Testing in Space Programs”, Crosslink, Volume6, Number3(Fall2005)

[2] Dima Zemskyy, MSE 575, Dr. Kornecki, Safety and Reliability Considerations in DO 178B, Fall 2006

[3] FAN Linbo, WU Yingcheng, ZHAO Ming, DAI Bifeng Analysis and Proof of the Differences between Software Reliability and Safety，2008

[4] N.G.Leveson. "A New Approach to System Safety Engineering", Aeronautics and Astronautics, Massachusetts, Institute of Technology, Draft of New Book, 2005.

[5] N.G.Leveson. "The Role of Software in Spacecraft Accidents", AIAA Journal of Spacecraft and Rockets, Vol. 41, No. 4, July 2004.

[6] GJB900 General Program for System Safety

[7] ARP4761 GUIDELINES AND METHODS FOR CONDUCTING THE SAFETY ASSESSMENT PROCESS ON CIVIL AIRBORNE SYSTEMS AND EQUIPMENT
[8] NASA-GB-8719.13 NASA Software Safety Guidebook
[9] MIL-STD 882C System Safety Program Requirements
[10] Mats P.E. Heimdahl, Safety and Software Intensive Systems: Challenges Old and New, IEEE
[11] Lahoz C.H.N, Camargo Jr.J.B. Abdala, M.A.D, Burgareli L.A, A Software Safety Requirements Elicitation Study On Critical Computer Systems. IEEE
[12] Robyn Lutz, Software Engineering for Safety: A Roadmap, IEEE
[13] P. V. Bhansali, Software Safety: Current Status and Future Direction. ACM SIGSOFT Software Engineering Notes Page 1 January 2005 Volume 30 Number 1
[14] ZHANG Tong Wei, SHI Zhu, Applications of WL_Net to the Missile Control Software Safety Analysys，Aerospace Control Vol.25,No.2,Apr.2007.
[15] Joint Software System Safety Committee, SOFTWARE SYSTEM SAFETY HANDBOOK
[16] GJB/Z 102-97, Software Reliability and Safety Design Criteria
[17] GJB/Z 142-2004, Guide for Military Software Safety Analysis
[18] Sidney L. Smith and Jane N. Mosier, GUIDELINES FOR DESIGNING USER INTERFACE SOFTWARE
[19] D.S. Herrmann, Software Safety and Reliability, (Los Alamitos, CA: IEEE Computer Society, 1999)
[20] System Safety Analysis Handbook, 2nd Ed, System Safety Society, July 1997.
[21] Alan C.Tribble, David L.Lempia, Steven P.Miller, Rockwell Collins, Cedar Rapids, Iowa Software Safety Analysis of A Flight Guidance System. IEEE