Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 80 (2014) 153 – 164

Procedia Engineering

www.elsevier.com/locate/procedia

3rd International Symposium on Aircraft Airworthiness, ISAA 2013

Safety analysis of software requirements: model and process

Shaojun Li[a]*, Suo Duo[a]

[a]*China Aero-poly technology Establishment, AVIC, Jingshun Rd7, Chaoyang District, Beijing 100191, China*

Abstract

Since lots of software hazards are caused by incompleteness or omissions of requirements, safety analysis of software requirements becomes more and more important. However, there are no systematic researches and exposition on the scope and the requirements of safety analysis. Safety analysts often get confused about how to take a complete analysis of software safety requirements. In this paper, referring to the software V&V model, an analysis model is proposed to specify the scope of software requirements safety analysis. Then, a process framework of safety analysis is determined to verify those analysis requirements derived from the analysis model. Relevant software safety analysis techniques which may be used in the analysis process framework were classified. Finally, the analysis model and process were applied to the landing gear control system.

*Keywords:*Safety; software safety; safety-critical software; software safety analysis; software requirements analysis

1. Introduction

With the increasing complexity of modern systems, the scale and complexity of software and its proportion of function in the entire system rise sharply. Software plays an increasingly important role in the operation and control of hazards, as well as in the safety-critical functions [1]. Failures of safety-critical software may cause serious damages to the equipment or properties, and even threatened the lives of persons.

Analysis of safety-critical software is an important means to recognize system risks and eliminate the hazard reasons, especially in the requirements phase. For reasons of incomplete and incorrect definition of the software safety requirements, a number of safety flaws are brought into systems early in the

* Shaojun Li. Engineer. Tel.:+0-135-8160-5349
E-mail address:jinerli@126.com.

doi:10.1016/j.proeng.2014.09.071

development period which are difficult to recognize later or expensive to modify [2]. Therefore it is necessary to take safety analysis of software requirements as soon as possible. A series of standards and manuals have been developed to conduct safety analysis, as well as the special analysis methods. However, there are no systematic researches and exposition on the scope and the requirements of safety analysis. Safety analysts often get confused about how to take a complete analysis or whether all the safety defects are recognized. Meanwhile, there is no complete analysis process leading to identify all defects that may cause safety incidents, either.

In order to solve the problems above, a safety analysis model of software requirements was built in this paper to determine the analysis scope and analysis requirements. Meanwhile, a complete analysis process was put forward to cover all the analysis requirements, which made it clear how to carry out safety analysis in the requirements phase. Also, relevant analysis techniques that can be used to support the analysis process were classified. At last, a case was studied to verify the effectiveness of the analysis model and analysis process.

2. Model for safety analysis of software requirements

2.1. Model for analysis of software requirements

IEEE Std 1012-2004（IEEE Standard for Software Verification and Validation）[3] was published by Software Engineering Institute, which belongs to the subject of System Engineering. As a process standard, it involves the whole software life cycle, and it is compatible with all life cycle models. In the IEEE Std 1012-2004，Verification & Validation (V&V) contain three dimensions which are *Process*, *Activity* and *Task*. For Requirements V&V which belongs to the Activity dimension, its model is shown in Fig.1.

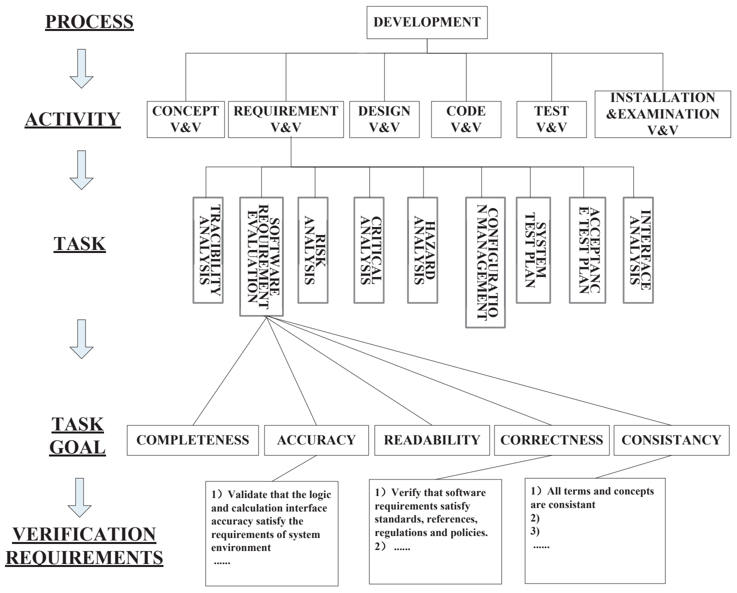
PROCESS
DEVELOPMENT
ACTIVITY
CONCEPT V&V
REQUIREMENT V&V
DESIGN V&V
CODE V&V
TEST V&V
INSTALLATION &EXAMINATION V&V
TASK
TRACIBILITY ANALYSIS
SOFTWARE REQUIREMENT EVALUATION
RISK ANALYSIS
CRITICAL ANALYSIS
HAZARD ANALYSIS
CONFIGURATION MANAGEMENT
SYSTEM TEST PLAN
ACCEPTANCE TEST PLAN
INTERFACE ANALYSIS
TASK GOAL
COMPLETENESS
ACCURACY
READABILITY
CORRECTNESS
CONSISTANCY
VERIFICATION REQUIREMENTS
1）Validate that the logic and calculation interface accuracy satisfy the requirements of system environment ......
1）Verify that software requirements satisfy standards, references, regulations and policies.
2）......
1）All terms and concepts are consistant
2)
3)
......

Fig. 1 Requirements V&V Model

As shown in Fig.1, Requirements V&V contains V&V tasks such as traceability analysis, software requirements evaluation, hazard analysis etc. Goals of requirements V&V are to guarantee the correctness, completeness, accuracy, testability and consistency. In order to satisfy these goals, detailed verification requirements are proposed. According to this model, after determining the Requirement V&V tasks, detailed verification requirements could be derived according to decomposition of different goals such as correctness, consistency, etc. Tasks of safety analysis of software requirements could be specified according to the tasks of software requirements V&V model. Relevant safety analysis requirements could be gathered through decomposition of different goals.

2.2. Model for safety analysis of software requirements

According to Fig.1, task dimension should be determined first to build the model of software requirement safety analysis. Since analysis of software safety requirements is part of analysis of software requirements, tasks of safety analysis should cover the tasks of software requirements analysis.

Initial software safety requirements come from system safety requirement, and flow into different subsystems. Meanwhile, other safety requirements, derived from bottom-up analysis, are flowed up from subsystems and components to the system level requirements. Problems in the flow-down process can be caused by incomplete analysis, inconsistent analysis or inexperienced analysts [2]。Therefore, traceability analysis should be taken as part of software safety requirements analysis. Software safety requirements should be placed into a tracking system to ensure traceability of software safety requirements throughout the software development cycle from the highest level specification all the way to the code and test documentation. In order to guarantee the traceability, correct and complete identification of safety requirements is the basis, which is the goal of requirements criticality analysis。

Safety-relevant software features, functions, performances and interfaces all could contribute to safety defects. Safety requirements should not only contain task-related or non-task-related functional, data, and interface requirements, but also performance, accuracy, robustness, capacity and other quality requirements [4]. Therefore, safety analysis of safety-related software requirements and interface requirements is also one task. Hazard analysis is used to ensure appropriate treatment, control and remission, meanwhile, hazard analysis could find if new hazards are brought in.

As above, tasks of software safety requirements analysis contain traceability analysis, criticality analysis, safety-related requirements analysis(including function and performance analysis), safety-related interface and data analysis, hazard analysis. Analysis requirements could then be determined through decomposition of different tasks according to the correctness, consistency, completeness, accuracy, testability and readability. Finally, the model for software safety requirements analysis is proposed as shown in Fig.2. Since the state machine could be used to take brief and full description of completeness requirements [5], state machine is used to describe completeness requirements of different analysis tasks.

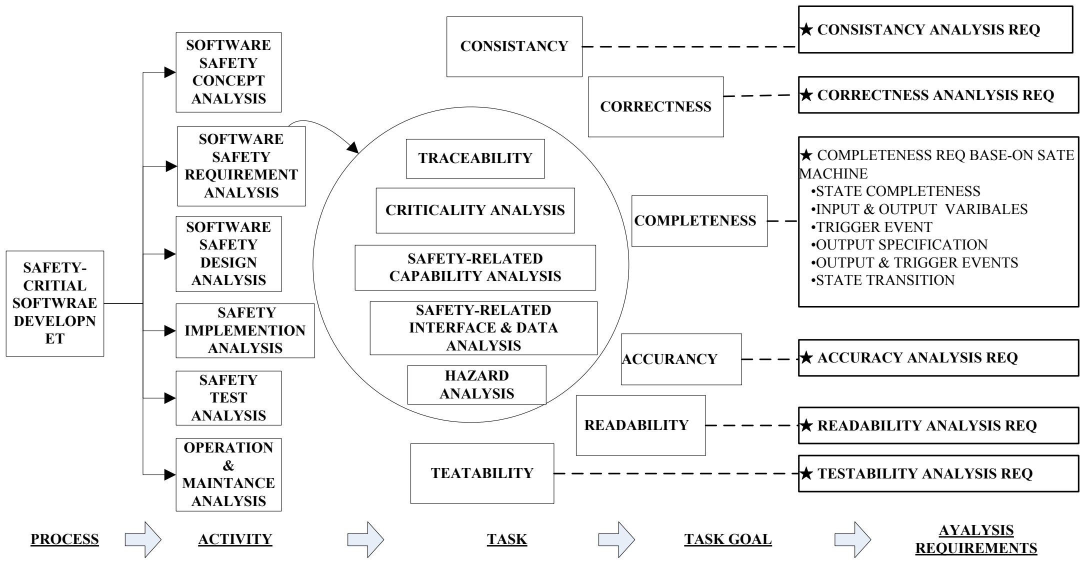
SAFETY-CRITIAL SOFTWRAE DEVELOPNET
SOFTWARE SAFETY CONCEPT ANALYSIS
SOFTWARE SAFETY REQUIREMENT ANALYSIS
SOFTWARE SAFETY DESIGN ANALYSIS
SAFETY IMPLEMENTION ANALYSIS
SAFETY TEST ANALYSIS
OPERATION & MAINTANCE ANALYSIS
TRACEABILITY
CRITICALITY ANALYSIS
SAFETY-RELATED CAPABILITY ANALYSIS
SAFETY-RELATED INTERFACE & DATA ANALYSIS
HAZARD ANALYSIS
CONSISTANCY
CORRECTNESS
COMPLETENESS
ACCURANCY
READABILITY
TEATABILITY
★ CONSISTANCY ANALYSIS REQ
★ CORRECTNESS ANANLYSIS REQ
★ COMPLETENESS REQ BASE-ON SATE MACHINE
•STATE COMPLETENESS
•INPUT & OUTPUT VARIBALES
•TRIGGER EVENT
•OUTPUT SPECIFICATION
•OUTPUT & TRIGGER EVENTS
•STATE TRANSITION
★ ACCURACY ANALYSIS REQ
★ READABILITY ANALYSIS REQ
★ TESTABILITY ANALYSIS REQ
PROCESS
ACTIVITY
TASK
TASK GOAL
AYALYSIS REQUIREMENTS

Fig. 2 Model for software safety requirements analysis

After gathering relevant analysis requirements relied on experience and references, 118 analysis requirements are collected, and they are classified into five checklists to conduct analysis later. The checklists are shown in Fig.3.

2.3. Completeness requirements of software safety requirements analysis

Accident model based on system theory believes that accidents are caused by interaction among different components which constitute the system. Safety is an emerging character occurred during the interaction. Emerging characters are controlled by a serious of constraints about the system behavior. When software dose not force necessary and appropriate constraints on the system behavior or system behavior deviates the constraints, software accidents would occur. Therefore, in order to avoid emerging of accidents, complete requirements or constraints should be detailed described in the software requirements or specifications.。

As the abstract of software requirements, state machine could be used to describe controlling behaviors of computers easily. Many specifications are specified by state machine. Safety especially relies on the completeness and accuracy of software internal process models. Specification written via the state machine could clearly describe this process model and relevant functions. Completeness analysis is to ensure the process model adopted by software is complete and accurate enough to prevent the occurrence of hazardous state.

For each state machine, following items should be restrained with safety rules: state. Input and output variables, relations between triggers and relevant outputs, state transition. Fig.4 shows completeness constraints for state machines.

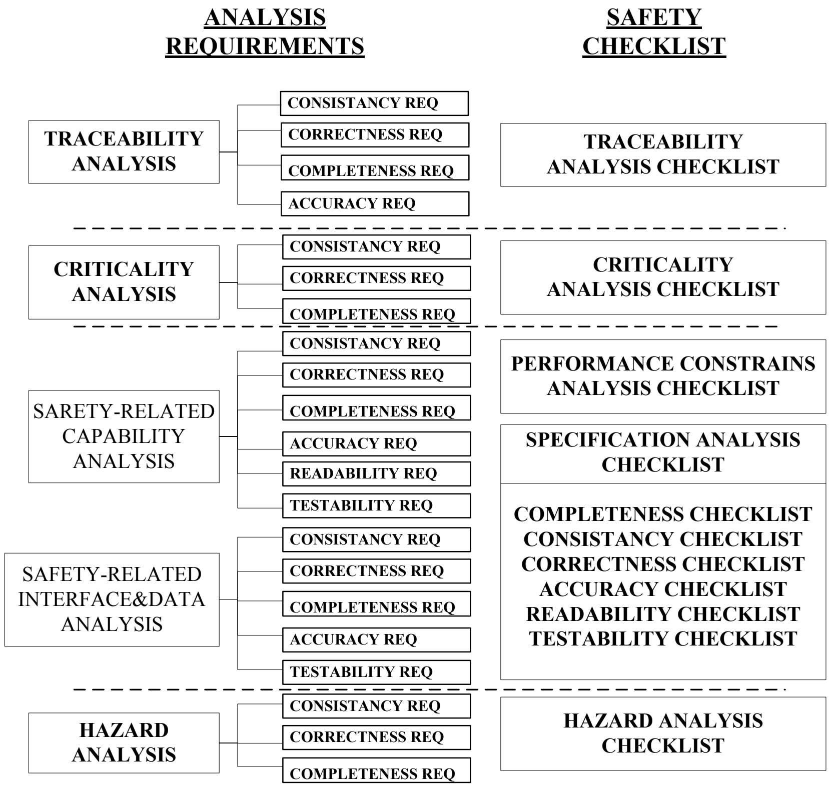

Fig. 3 Checklist of software safety requirements analysis

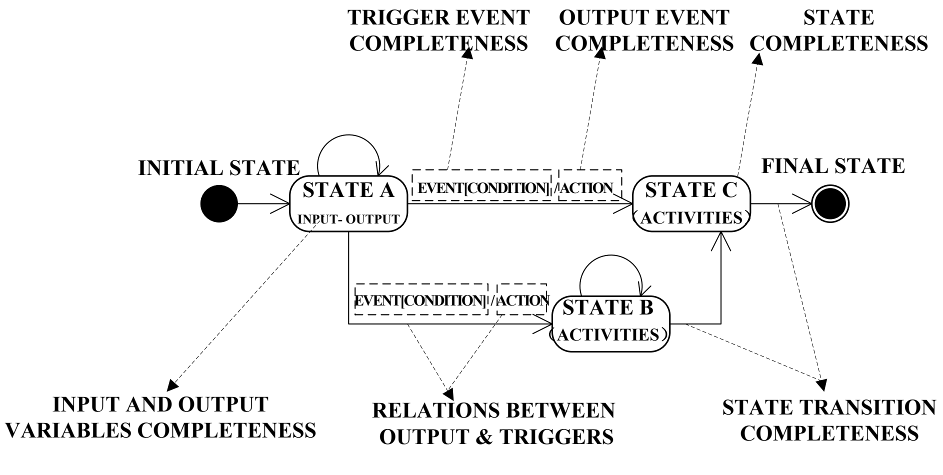
TRIGGER EVENT COMPLETENESS
OUTPUT EVENT COMPLETENESS
STATE COMPLETENESS
INITIAL STATE
STATE A
INPUT- OUTPUT
EVENT[CONDITION]
/ACTION
STATE C
ACTIVITIES
FINAL STATE
EVENT[CONDITION]
/ACTION
STATE B
ACTIVITIES
INPUT AND OUTPUT VARIABLES COMPLETENESS
RELATIONS BETWEEN OUTPUT & TRIGGERS
STATE TRANSITION COMPLETENESS

Fig. 4 Completeness constraints for state machines

Relying on experiences and relevant references [6][7][8][9], 76 safety rules are collected as shown in Table 1.

Table 1 Completeness rules on state machines

Item	Sub Item	Number
State completeness	Relations between modes and functions, start completeness, degradation mode design, interlock design, fault-tolerant design	18
Input & Output variables	Input & output variables	2
Trigger event completeness	Robustness, certainty, essential value and timing assumption, capacity or load, hazardous commands	34
Output specifications completeness	Environmental capacity considerations, data periods, delays	8
Relations between output and trigger events	Feedback circuit	3
State transition completeness	State transition	11

3. Process of software safety requirements analysis

3.1. Process framework of sofware safety requirementsanalysis

A process framework of software safety requirements analysis should be established to verify these analysis requirements in Chap. II. When establishing the process framework, analysis procedures that belong to the process framework should be specified first. And then, in order to take a step-down and iteration analysis, and make the previous analysis results support the later analysis procedures, the sequences of each analysis procedure should be specified. Finally, the process framework is built as shown in Fig.5.

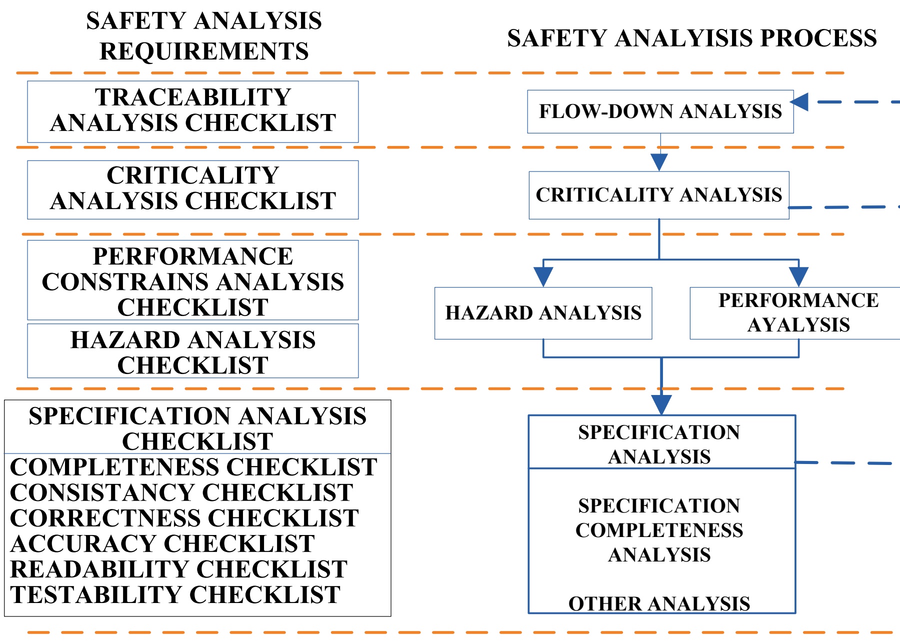

Fig. 5 Process framework of software safety requirements analysis

1) Requirements flow-down analysis

Requirements flow-down analysis is to verify if system safety requirements and hazard analysis results flow correctly, completely and accurately down to the software safety requirements.Since transforming the system safety requiremetns to the softare safety requirements is first step of software safety engineering, flow-down analysis should be taken first. A tracing matrix of safety requirements is a good way to take tracibility analysis.

2) Safety criticality analysis

Safety criticality analysis tries to verify if all safety-related requirements and relevant safety levels are clearly identified through analyzing all software requirements and their relationships. Thus, it could be ensured that safety requirements recorded in the tracing system are complete and correct. Through criticality analysis, the safety requirements tracing matrix could be updated. Meanwhile, safety criticality analysis could clearly list safety-related requirements, which is helpful to specify the emphasis of later safety analysis and determine the starting point of safety design and coding. Therefore, criticality analysis builds a connecting link between the preceding and the following analysis procedures.

3) Hazard analysis

After determining safety-critical software requirements, function design is further detailed, and new hazards or hazard causes may be introduced in. Therefore, it is necessary to take hazard analysis. Hazard analysis is the basis of building a complete state machine (containing all hazardous states) for the following specification completeness analysis. The results of hazard analysis could be used to update the traceability matrix and the criticality analysis.

4) Performance constraints analysis

Performance analysis and hazard analysis could be taken at the same time. Performance constraints analysis evaluates software requirements that relate to execution time, I/O data rates and memory/storage allocation. This analysis focuses on program constraints. Typical constraints requirements are maximum execution time, maximum memory usage, maximum storage size for program, and I/O data rates the program must support. Performance constraints analysis belongs to subject of software performance engineering, and it’s independent of hazard analysis. The results of Performance analysis could be used to update the traceability matrix and the criticality analysis.

5) Specificaion analysis

Specification analysis is to verify the completeness, correctness, consistency and testability of software specification. Specification analysis needs results of hazard analysis and performance analysis. For example, in order to verify there is no hazardous path for specification completeness analysis, the results of hazard analysis are necessary. Performance analysis could help better understanding of system properties (such as timing properties). Therefore, specification analysis is the last step. Completeness analysis of specification is the most important part. We recommend that using the state machine and completeness checklist to take completeness analysis.

3.2. Software safety requirements analysis techniques

Software safety requirements analysis techniques could be classified according to the analysis process. Table1 summaries the techniques could be used in different analysis procedures.

Table 2 Analysis techniques classification

Analysis Process	**Relevant analysis techniques**
Data-flow analysis	Checklist, tracing matrix, formal methods
Criticality analysis	Checklist, criticality matrix
Specification analysis	Informal：reading analysis, traceability analysis, checklist
	Semi- formal :logic/ function diagram, sequence diagram, data flow , FM/state transition, time petri nets, judge/truth table, control-flow, information-flow, function simulation
	Formal: formal specifications, model checking
Performance constraints analysis	Estimate according to similar systems, performance modeling, simulation
Hazard analysis	Traditional: FTA, FMEA, HAZOP, SDA
	Modern: FPTN, HIP-Hops, FSAP-NuSMV, Altarica, etc.

4. Case Study

4.1. Introducntion of landing gear control sysytem

The landing gear putting up and down system has two similar control units, which adopts dual-redundancy design. Taking the hand shank position signals and signals collected by sensors about plane position as inputs, control units control the putting up or down of landing gears and wheel doors according to the sequential logic. Meanwhile, the control units provide other functions such as position indication of wheel doors and landing gears, rationality monitoring and warning. When control unites find internal failures, the working control unit switch from the main control unit to the redundant unit. The function diagram is shown in Fig.6.

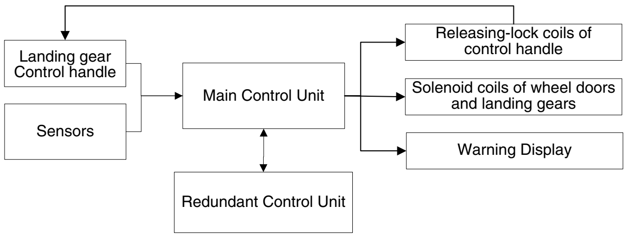
Landing gear Control handle
Sensors
Main Control Unit
Redundant Control Unit
Releasing-lock coils of control handle
Solenoid coils of wheel doors and landing gears
Warning Display

Fig. 6 Function diagram of landing gear control system

The landing gear control system is a typical safety-critical system, because that its failures would trigger catastrophic accidents. For example, put up landing gears in the ground glide phase; can’t put down landing gears. Therefore, safety analysis of landing gear control system should be taken to find and eliminate safety defects.

4.2. Process of software safety requirements analysis and results

1) *Requirements flow-down analysis*

A safety requirements flow-down traceability matrix is built from softwrae commitments to software specification as shown in Table 3.

Table 3 Requirements down-flow traceability matrix

Number	Safety requirements in software commitments	Safety requirements in software specification
1	Real time response to landing gear control system	1.1 real time response to putting up control
		1.2 real time response to putting down control
2	Failure gathering and handling	2.1 failure detection logic
		2.2input rationality monitoring
		2.3failure warning logic
		2.4failure handling logic
3	Fault-tolerant design	3.1redundancy voting logic
		3.2redundancy switch logic
4	…….	……

Trough traceability analysis, the inconsistent design logic between these two documents could be found. For example, the rationality judgment logic of input signals of landing gear control handle is described in software commitments as follows:

$$X = A + B \tag{1}$$

However, in the software specification, the irrationality judgment logic is described as follows:

$$\overline{X} = \overline{(A + B)} \tag{2}$$

The inconsistency of the judgment logic in two documents could be found easily.

2) *Requiremetns criticality analysiss*

Criticality matrix is built to identify all safety-related requirementsas shown in table 4.

Landing gear systems control the safety-critical functions independently and own a certain extent of complexity. Moreover， some hazard control procedures are time-critical (such as putting down the landing gears). Therefore, the landing gear control systems belong to Category I control software [2]. The loss of this control system would cause a catastrophic accident [10]. Above all, the safety index or hazard index of landing gear control system should be cataloged as Level I (or Level A). However, system designers define the software as Level B, and follow the management and verification requirements of Level B, which is incorrect.

Table 4 Requirements down-flow traceability matrix

Mission Control Functions	**Hazard**		
	Uncommand putting up landing gears	Failed in putting down landing gears	…
Release lock of handles	×	×	
Activate putting up function	×		
Putting up control of landing gears	×		
Putting down control of landing gears		×	
……	……	……	…...

3) Performance constraints analysis

The landing gear control system is improved based on previous models. The *functions* of two systems are mainly the same, but the hard portion of the improved system owns better performance. Therefore, after estimating based on similar systems, the hardware portion and software portion composing this system could satisfy the performance requirements.

4) Hazard analysis

Such failures of the landing gear control system would cause aircraft-level hazards: uncommanded retracting landing gears(catatrophic, specially on ground). Can't put down landing gears (catatrophic)、*putting* down landing gears at a high speed in the air [10]. According to fault tree analsys, we find that there are no relevant measures to eliminate some minimal cuts of the top events, such as, a wrong putting down command because of swith failures, mistaken operation, failures of external devices of control units, internal failures of software, etc.

5) *Specifiction analysis*

Completeness analysis is the key of specification analysis. In this case, completeness analysis was conducted through tailoring completeness checklists, building state diagrams of UML and *model* checking relying on the checklists(manually or automatically). Through completeness analysis, 18 safety defects were found. Partial results are shown in Table 5.

Table 5 Results of completeness analysis

Analysis Req. based on State Machine	**Safety defects (18)**	**Typical defects**
State completeness	4	The state constraints " it is forbidden to put down landing gears at a high speed in the air", "it's forbidden to retract landing gears when the aircraft is landing off" are not declared in the software requirements, which lead to there are no relevant designs to eliminate the hazardous states.
Input & Output variables	1	In the specification, logic "0" is represented by "0110", and the logic "1" is represented by "1001". However, there are no treatment measures to handle the occurrences of "1000" and "0011".
Trigger event completeness	8	There is no compulsory way to switch working control unit from the main unit to the redundant unit if the switch function failed.
Output specification completeness	1	The data age of control variables (such as retracting or putting down landing gears) is not declared. In the take-off run, the retracting command would not be executed, but it would still kept effective since there are no constraints on data age.
Relations between output and trigger events	0	No defects found.
State transition completeness	4	When the landing gears are put down, the landing gears retract function couldn't be activated if the sensors on the tail wheel gears failed. Then, the landing gears couldn't be retracted. There is no relevant solution in the specification to deal with this situation.

5. Conclusion

In order to identify all potential hazards in software safety requirements and specify the whole scope of software safety analysis, a software safety requirements analysis model is put forward in this paper. The completeness analysis requirements are proposed based on the state machine theory, which shows to be a good way to specify the constraints on software processes. An analysis process framework is also put forward to very these analysis requirements, which is proposed according to different checklists. Relevant techniques are also classified for the analysis process. The case study of landing gears control system shows that the model and analysis process are useful to verify the safety of software requirements.

Acknowledgements

The work descried in this paper was with the great help of Professor Xiaohong Bao, whose advices had a great effect on the accomplishments of this research.

6. References

[1] J. Hill,S. Tilley. Creating Safety Requirements Traceability for Assuring and Recertifying Legacy Safety-Critical Systems[M]IEEE，2010: 297-302

[2] NASA-GB-8719.13-2004. NASA Software Safety Guidebook[S]．National Aeronautics and Space Administration, 2004

[3] IEEE Standard for Software Verification and Validation[S], IEEE, Std 1012, 2004.

[4] D. G. Firesmith. A Taxonomy of safety-related requirements[J]. Requirements Engineering. 2004

[5] N.G. Leveson, Safeware: System Safety and Computers[M], Addison-Wesley, Reading,MA, USA, 1995.M.

[6] M. S. Jaffe,N. G. Leveson,M. P. E. Heimdahlet al. Software requirements analysis for real-time process-control systems[J]. Software Engineering, IEEE Transactions on. 1991. 17（3）: 241-258

[7] NASA-STD-8719.13B NASA software safety standard [S].2004.

[8] N.G. Leveson, Safeware: System Safety and Computers[M], Addison-Wesley, Reading,MA, USA, 1995.M.

[9] M. S. Jaffe,N. G. Leveson. Completeness, Robustness, And Safety In Real-time Software Requirements Specification[M]，1989: 302-311

[10] Dewi Daniels. Developments in Risk-based Approaches to Safety[M]. Bristol, UK, pp 199-213,2006