ISSN No. 0976-5697

Volume 5, No. 5, May-June 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Safety-Critical System and Testing the Integrity Software

Mina zaminkar
Department of Computer Science
College of engineering, Yazd Science and Research Branch,
Islamic Azad University, Yazd, Iran

Mohammad R. Reshadinezhad
Department of Computer engineering
University of Isfahan
Isfahan, Iran

Sima Emadi
Department of Computer
College of engineering, Yazd Science and Research Branch,
Islamic Azad University,
Yazd, Iran

Abstract: Safety is a characteristic of a system that guarantees the human life, environment and organizational entities against hazards; in a system designed with the objective of approaching a level of coherent safety with executing all of its necessary functions. This software testing is concerned with assessing the analysis results, programs' behavior and the conformity of the program with the safety requirements. Here the safety system, reducing the hazards to a minimum and conducting different tests necessary for making the software coherent will be described.

Keywords: Safety-critical systems; Hazard analysis; Software testing; Hazard reduction; Software risk.

I. INTRODUCTION

Having a safe system is a team effort and safety is everyone's responsibility in industrial companies today. Software is a vital part of most systems and it controls hardware and provides mission-critical data. Software must be safe in order to be profitable for any company [1]. But how can one say that a software is "safe" or "unsafe"? What are the hazards that software may contribute to, or that software may control? Why should one care about software safety?

When a device or system can cause injury, death, loss of vital equipment, or damage to the environment, system safety is paramount. The system safety discipline focuses on "hazards" and the prevention of hazardous situations. Hardware or software that can lead to a hazard, or is used to control or mitigate a hazard, comes under that category. Software has become a vital and integral part of most systems. Software can respond quickly to potential problems, provide more functionality than equivalent hardware [2]. The software safety discipline spread out beyond the immediate software used in hazard control or avoidance to include all software that can impact hazardous software or hardware. All such software is "safety-critical" [1]. Systems engineers, Project managers, software engineers, software assurance personnel, and system safety personnel all play a part in creating a safe system. A software is safety-critical if it performs any of the following:

a. Controls hazardous or safety-critical hardware or software.
b. Monitors safety-critical hardware or software as part of a hazard control.
c. Provides information upon which a safety-related decision is made.
d. Performs analysis that impacts automatic or manual hazardous operations.
e. Verifies hardware or software hazard controls.
f. Can prevent safety-critical hardware or software from functioning properly.

Safety-critical software includes hazardous software (which can directly contribute to, or control a hazard). It also includes all software that can influence that hazardous software [2].

In the past, hardware controls were the primary method used to control/prevent hardware hazards. Today, because of the complexity of systems, it may not be feasible to have only hardware controls, or to have any hardware controls at all. Now, many hardware hazard causes are addressed with software hazard controls. Often this is because of the quick reaction time needed to respond to a failure or the complexity of detecting possible faults and errors before they become failures.

A fault is any change in the state of an item which is considered anomalous and may warrant some type of corrective action. A failure is the inability of a system or component to perform its required functions within specified performance requirements.

a) A fault may or may not lead to a failure.
b) One or more faults can become a failure.
c) All failures are the result of one or more faults.

Fault tolerance is the ability of the system to withstand an unwanted event and maintain a safe and operational condition. It is determined by the number of faults that can occur in a system or subsystem without the occurrence of a failure. Fault and failure tolerance are often used synonymously, though they are different.

A System Safety Program Plan is a prerequisite to performing development or analysis of safety-critical software. The System Safety Program Plan outlines the organizational structure, interfaces, and the required criteria for analysis, reporting, evaluation, and data retention to provide a safe product. This safety plan describes forms of analysis and provides a schedule for performing a series of these system and subsystem level analyses throughout the

development cycle. It also addresses how the results of safety analyses will be communicated and the sign-off/approval process for all activities. A Safety Program Plan is usually created and maintained at an organizational or "programmatic" level. Within NASA, a program may have one or many projects [3]. At the project level, there should also exist a safety plan which describes for that project how it will incorporate the programmatic plan requirements as well as those specific to the project [4].

Table I. Hazard Prioritization-System Risk Index

Probability					**Intensity Levels**
Impossible	**Unlikely**	**Possible**	**Probable**	**Likely**	
4	**3**	**2**	**1**	**1**	**Catastrophic**
5	**4**	**3**	**2**	**1**	**Critical**
6	**5**	**4**	**3**	**2**	**Average**
7	**6**	**5**	**4**	**3**	**Trivial**

Hence, a safety-critical software system could be defined as any system whose failure or malfunction can severely harm people's lives, environment or equipment. These kinds of risks are managed using techniques of safety engineering.

The aim of this paper is to provide a brief overview of safety-critical software systems and describe the main techniques or approaches used to design and test these kinds of systems.

This paper first introduces standards used and applied in different fields when developing safety systems. The next section focuses on the level of risk for safety-critical software systems. The paper then will go on to describe different approaches on Testing and integration of the software. Then, in the next section, it is illustrated that what kind of tests are of concerned. Finally, the last section contains the conclusion remarks.

II. THE SAFTY SYSTEM

The sensitive safety system is a system designed with the objective of reaching to a point of coherent safety where all the necessary functions of safety are designed in a manner where it would not become defective during implementation [5]. This system needs to have reliability and resistance against defect, error and damage [3].

Whenever software deals with the control and supervision of the hazards and or controls software, hardware, of sensitive-safety it is considered as sensitive-safety software. This software is usually installed in systems with remote-control and very high-speed capabilities: air pressure control of the chamber of the aircraft and strong laser control, the hazardous aspects. The fire alarm software is another type of sensitive-safety software [2] .The software which provides the data regarding decisions made on safety is in this category as well. In simple words, sensitive-safety system is the one if not processed properly it would be the cause for irreparable damages but if processed properly it can make the system run even if something happens. Hence, this is a highly functioning continuous accurate system which acts whether there is a default or not [1].

The major feature of this system is its reliable and resistant nature in withstanding defect and error. In order to have a reliable and resistant system a specific order should prevail in designing process, that is: hidden system engineering, protocol and network engineering, safety engineering, assurance engineering rapid response engineering and systems engineering [6].

Any part of a software controlled through hardware, software or human operator is considered as the potential factor in hazard that is the sensitive-safety software and is tested against quality control and analysis; assessed through the software safety analysis until their final approval. The necessities of sensitive safety must be designed to guarantee the changes which may occur in the future [7].

Implementation of a PHA system is the prerequisite in applying a system in hazardous environments through the analysis of software [2] with the initial outcome of the PHA system at hand, the safety requirements are obtained and the hardware and software requirements are determined. When the specifications of the system's design are known as the hazard analysis of the sub-systems and the system components can begin to operate. The PHA is the first source of specific safety system requirements and possibly can advance up to the specific safety software requirements (unique to specific systems' architecture). This issue, accompanied with the general hazard defined for the system is considered as a prerequisite in implementing any type of safety software analysis [8].

When implementing the PHA the manner in which the software interacts with other parts of the system is a vital point. The software is the heart and the brain of many complicated systems of today, controlling and supervising almost all the functions. When the system is analyzed through marginal elements the manner in which the software interacts with other sections must be considered [8].

The PHA should be aware of the systems sections supervision manner with respect to the software (a defeated sensor would be subject to inappropriate response of the software).

III. LEVEL OF RISK

Hazard analysis, like the PHA is not involved whether there are the potential of hazard occurrence. All hazards are considered bad even with the least of occurrence chance. Though, usually there is not enough money and time to evaluate all potential hazards, the hazard should somehow be prioritized. This prioritization would lead to the conceptualization of hazard [2]. The System Risk Index, based on the above severity levels and likelihood of occurrence, is shown in Table I. In this table, 1 is the highest and 7 is the lowest priority. Each program, project, or organization should create a similar risk index, using their definitions of severity and likelihood [9], [10].

Hazard is a combination of the following two issues:

a. The probability (quantitative/qualitative) of whether a plan or project may be subject to an unwanted event: safety incident, safety agreement and or system component failure
b. The consequences, effects or the intensity of the unwanted event that may cause the hazardous event to take place

Every project or plan, by referring to the regulated definitions in the procedures, standards and policies of the Agency is in need of a collection of definitions with respect to "Hazard intensity" level. The selected definition should fit the organizations scope of activities.

When the team members of different disciplines discuss about the causes and the control of hazards involving software, a common language is essential to assist the mutual understanding [10].

Prioritization of hazards is essential in order to determine the source which is ranked at level 1, which is not allowed in systems design. Any system which faces the "1" the hazard index must be reviewed completely for elimination or at least reduction of the potential in hazard occurrence. The lowest index is "5" though there exist "6" and "7" as well. For levels 2, 3 and 4, the safety analysis level requirement and hazard level is high and is shown in Table II.

Table II. Prioritization of hazards

Class safety practices recommended	Risk index system
Inapplicable(forbidden)	**1**
Full	**2**
Average	**3**
At least	**4,5**
Optional	**6,7**

A. Hazard elimination:

The hazards are eliminated as far as possible and it's best to do this in a designed manner just like eliminating an energy source. For instance, software can influence the pressure control while there is no need to for the software to have access to the control; therefore, an error in software function could lead to hazard occurrence. Thus, prevention of software access to the control would nullify the possible involvement of the software in hazard occurrence. With respect to system, hazard elimination is conducted in a manner where a solution is found in design where there is no need for high hazard pressure.

B. Hazard reduction design:

The hazards cannot be eliminated completely, but they can be controlled. The PHS can assess what might be the cause of a hazard and provide the manners by which that hazard can be controlled, of course through a design. Hazards can be minimized through providence of resistance to failure (increment in series or parallel in an appropriate manner), major safety differences or automated safety. For example, the software confirms all the assessments on the conditions prior to the commencing the rocket engine.

IV. TESTING AND INTEGRATION OF THE SOFTWARE

The different tests that can be conducted are:

a. Immerging Test
a) Integrated and unit test
b) Integration of the software through hardware
b. System test
a) Practical
b) Functional
c) Loading
d) Stress
e) Incidents
f) Consistency
g) Acceptance
h) "red team"

It should be noted that the "system test" is not a single test, but a collection of possible executable tests. These resets are conducted by applying the whole system (hardware and software) although it is possible to use a simulator under specific circumstances. The system testing occurs when all the capabilities are tested, restrictions are identified and the strengths for resisting against errors and failures are realized well.

One of the most essential tools for the examiners, developers, and project managers is having a high perspective on all testing activities of system (including the sub-systems). It is possible to list these data with the following at the top (who is the responsible person when the incident takes place, what is the objective of the test, what are the features of the data involved in the test, the environment where the tests are conducted, what are the product and which are the input and output criterion) in a Table (as a wall chart) where all different levels of the test are illustrated.

A. Testing:

Testing the operational implementation is a portion of a software test in a real or simulated or environ. Testing involves the assessment of the results obtained from the analysis, assessing the programs' behavior and assessing the conformity of programs' safety requirements. Testing the software and the unit level (the integration and the system) is usually conducted by someone other than the producer (with exception of smaller drums) [2], [9].

Usually, testing the software confirms that this software has performed its functions accurately and is able to expose the praised behaviors to the interested. Safety test concentrate on the tests made on the weak points of the program and identification of the extreme or unexpected states which may cause S/W failure. This is a complementary test not a repeated expanding test. One sample of the techniques applied in specific safety test is injection of S/W error. This program is for the sensitive-safety software test as in Bart in San Francisco or the security and verification test of COTS tools. The errors are entered as codes before the test begins and then the answers are observed. Moreover, all of the practical restrictions and requirements must be subject to test at the bottom and the top of the announced restrictions.

B. Testing the unit level:

This test is programmed in the course of designing the details, that is, when the operations within a unit are already defined. These tests are conducted after the codes are compiled. Since it is possible that a code might have a defect, by repeating each section anew, the tests would be implemented once more. This test is conducted by the producer with the possibility that another producer in this test group may conduct the unit test. The major entry criteria for testing this unit is: each unit be compiled with no error.

The unit surface test is essential, since it can have access to levels of the software that possibly were inaccessible when the units become integrated. When the unit becomes integrated with the system, testing the input data of a unit might not become possible. The unit surface test can identify the difficulties in the lay-out and problems related to the function that would implement changes in their S/W. The faster the identification of the issues, lower the change costs. In case this surface is necessary for assessing the code, for each sensitive-safety unit the specific safety tests must be designed. These tests must illustrate the areas that are covered by the test requirements. Every safety test must have the defect and intact criteria. In the test design or software design the management should explain the assessment manner and the reports regarding the details of the sensitive-safety and the relevant problems thereof. This test report must be available for sections of the sensitive-safety units [2], [9].

The unit level tests are of two: the white box and the black box. The white box test includes items that the inner elements details of which are known. The black box test merely controls the input/output elements with no concern about what occurs inside the unit. The white box test is involved with items like assessing the route and the areas under the branches' coverage, chain implementation and implementation orders; while the black box is involved in the areas related to inputs (volumes), output and error elimination. The safety test must be concerned with proper implementation of all the safety requirements.

The actual safety tests can be implemented in an independent manner or as a section of the major testing activities of production.

The integrated test is mostly conducted in simulating environments. The system that usually is run on the real hardware. It is suggested for the hazardous tests to be run initially in simulating environments. No one likes to face accidents during commencing the rocket engine or turning of military apparatus. All defaults observed during tests should be analyzed and the procedure should be recorded in the differences reports and briefings regarding the tests. This difference report must contain the advancing problems, suggestive solutions and the final result of the problem. These reports must be handed in after the hardware has reached a specific level of growth (the basic levels or at Beta version). The changes due to identified problems are usually sent for assessment and approval or rejection through the Control Page Change software. The reports on these problems must be traced and their managements should run a review on the old and new problems of the software [9].

V. WHICH TESTS ARE OF CONCERN?

The tests to be considered are as follow:

a. The white box test: based on the awareness of the inner rationale of the program codes, including the codes coverage status, braches, routs and circumstance.
b. The black box test: based on the requirements and capabilities and not involved with any awareness regarding the internal design or codes
c. Unit test: is the smallest scale of testing run on specific operations or parts of codes. This test is conducted through the programmers not the testers; since it needs accurate knowledge on the internal design and codes. This is not an easy task, unless the software has a very good architecture and design requiring high expertise.
d. Integrated incremental test: is the continuous test of a software when new capabilities are added to it. It is necessary that the different aspects of the software capabilities be adequately independent and operate separately by with respect to other sections of the program or the test be developed up to a needed point. This test is run by the programmers or the testers.
e. The integrating test: is involved in the combined sections of a program in order to determine their interactive accuracy in operations. The "sections" constitute portions of the codes' elements, a personal user's program, service receiver or provider of applied programs or the network and or other issues. This type of test is specific to service receiver or provider and distributive systems.
f. The function test: is of the black box test type equipped with the requirements of program function, this test is run by the tester, and this does not mean that the programmers do not need to check whether their codes work properly (something applicable at every stage of the test).
g. System test: is of the black box test type based on the general requirements' specifications. This test covers all combined areas of a system.
h. End to End test: is similar to the system test. It has the biggest test scale which includes testing the whole environ of the S/W by imitating the real world conditions such as interacting with a database, using the networks connections, interacting with hardware, applied programs and or other systems if necessary.
i. Soundness of mind test: is usually the initial objective f the test to determine whether the new software version can be applied for bigger objectives or operations. For example if the new software stops the system every 5 minutes, then the system speed drops, or the database is destroyed; therefore, the software is not in a desired condition and it cannot guarantee further tests.
j. Regression test: is a repeated test run after the corrections or for making changes in the software or its environs. Determining the number of the required tests is difficult especially at the ending phases of production process. Automated test apparatus can be applied for such tests in a specific manner.
k. Reception test: is the final test based on the customer/user's final specifications, and or based on the application through the end user during some limited time periods.
l. Loading test: is the heavy loading program testing like testing a website subject to a vast spectrum of loads for determining that at which points in time the system would response to contraction and or failure.
m. Stress test: this term is synonymous with the "load" and "stress", and is used in describing test like system's function under unusable and heavy loading, many reiterations of measures and specific entities, high volume of digital entries, complicated and expanded search in a database system etc.
n. Function test: this term is synonymous with the "load" and "stress". The "Function test" in its ideal form (and any other kind of test) is defined in the records and

documents or requirements and or quality assurance documents and or test programs.

o. Usage ability test: is to make the users as friends. This test is inert and relates to the objectives of the user or the end customer. In this test the users' interviews, surveys of video recordings from their meetings and other approaches are applied. The programmers and testers usually do not conduct this test.

p. Installation/omission test: is the complete or partial test on the promotion process of installation and or omission.

q. Marketing test: is run on the recovery of the system from the incident, the hardware failures or other drastic problems.

r. Marketing test: is run on the recovery of the system from the incident, the hardware failures or other drastic problems.

s. Adaptability test: is to test the software operational manner in specific hardware, software, agent systems and networks.

t. User acceptance test: determines the customer and or final satisfaction from the software

u. Comparison test: compares the strong and weak points of the S/W with that of the competing products

v. Alfa test: is the software test at the end stages. Minor changes in designs are due to this type of tests. The end users conduct these tests.

w. The foundation test: is run when the development and tests are completed and is to find the final problems and defaults before publication. The end users conduct these tests.

VI. CONCLUSION

Issues regarding safety are the essential prerequisite for the development where lack of supervision on safety next to lack of efficiency in systems management safety is the main causes of incidents occurrence. Many of the S/W engineering methods are interested to determine and prevent errors. In very big systems the removal of all defaults cannot be guaranteed. The reported problems must be traced and the management should be aware of the existing and previous problems regarding the software.

VII. REFERENCES

[1] I. I. P. Ltd., "An Introduction to Safety Critical Systems," 1997.

[2] Nasa Software Safety Guidebook, " Nasa Technical Standard," 31 March 2004.

[3] J. R. Pimentel, "Designing safety-critical systems: A Convergence of Technologies," Kettering University, Flint, Michigan, 2008.

[4] M. Zaminkar, M. R. Reshadinezhad, "A comparison between two software engineering process,RUP and Waterfall models," International Journal of Engineering Research and Technology, Vol. 2, Issue 7, July 2013.

[5] R. Ahmed, Y. H. Jeong, G. Heo, "Design of safety-critical systems using the complementarities of success and failure domains with a case study," www.elsevier.com, 2011.

[6] B. J. Krämer, N. Völker, "A highly dependable computing architecture for safety-critical control applications," Real-Time Systems, vol. 13, pp. 237-251, 1997.

[7] E. G. Leaphart, B. J. Czerny, J. G. D'Ambrosio, C. L. Denlinger, and D. Littlejohn, "Survey of software failsafe techniques for safety-critical automotive applications," SAE World Congress, Detroit, pp. 1-16, 2005.

[8] D.S. Herrmann : Software Safety and Reliability - Techniques, Approaches,and Standards of Key Industrial Sectors, ISBN 0769502997.

[9] P. H. Yacov, Y. Haimes , "Software Risk Management," Technical Report CMU/SEI-96-TR-012 ESC-TR-96-012.

[10] R. King "*Risk Management*," Business Sequence Diagram Publishing system, ISBN 0948672722, 2005.

Short Bip Data for the Authors

Mohammad R. Reshadinezhad He was born in Isfahan, Iran, in 1959. He received his B.S. and M.S. degree from the Electrical Engineering Department of University of Wisconsin, Milwaukee, USA in 1982 and 1985, respectively. He has been in position of lecturer as faculty of computer engineering in University of Isfahan since 1991. He also received the PhD Degree in computer architecture from Shahid Beheshti University, Tehran, Iran, in 2012. He is currently Assistant Professor in Faculty of computer Engineering of Isfahan University. His research interests are digital arithmetic, Nanotechnology concerning CNFET, VLSI implementation, logic circuits design, Cryptography and software engineering.

Mina Zaminkar She received her B.S. in computer engineering (software) from university of Dolat Abad, Iran in 2010.She got her M.S. degree in computer engineering at the department of computer and research branch, Islamic Azad University, Yazd, Iran in 2013. Currently teaches computer courses at Shahid Ashrafi Isfahani University, Isfahan, Iran. Her research interests mainly focus on computer software engineering, data mining, data bases and biometric. She is currently engaged in research involving railway interlocking and Quality Management.