A Methodology for Safety Critical Software Systems Planning

EHAB SHAFEI[1], IBRAHIM F. MOAWAD[2], HANY SALLAM[1],
ZAKI TAHA[3], MOSTAFA AREF[3]
[1]Operation Safety and Human Factors Department, [2]Information Systems Department,
[3]Computer Science Department
[1]Nuclear and Radiological Regulatory Authority, [2,3] Faculty of computer and information sciences, Ain Shams University
Egypt
ehab_vip@yahoo.com, ibrahim_moawad@cis.asu.edu.eg, salamhany@yahoo.com,
ztfayed@hotmail.com, aref_99@yahoo.com

Abstract:- Recently Safety Critical Software Systems (SCSSs) become essential part of many critical systems such as Nuclear Power Plants (NPPs), radiation therapy, aircrafts, and many medical devices. Although of the vital role of SCSSs in saving human life, environment, and properties, there is no generic methodology for developing such systems based on standards and guidelines. This methodology sets an integrated model that concerned about the safety of critical software systems as a component of the critical systems. It consists of the needed processes and operations required for developing SCSSs free of faults. This methodology ensures that SCSSs are developed using processes which based on appropriate standards and guidelines and can be accordingly certified. The objective of this methodology is to produce certified critical software systems that comply and conform to standards and guidelines. The methodology consists of three phases (safety planning and requirements phase, safety analysis phase, and design, implementation, and operation phase). This paper is going to focus on safety plans phase. The insulin pump system is applied as a case study on the safety planning and requirements phase.

Key-Words: - Safety Critical Software Systems, Safety Analysis and Requirements, Insulin Pump

1 Introduction

Critical software systems become an important factor in assuring the safe operation of many critical systems. Safety must always be considered throughout the overall critical systems not limited to software but extended to contain computer hardware, electronic/electrical hardware, mechanical hardware, and operators. Critical software faults are often caused by deficient requirements analysis and design faults.

Certification of SCSSs is the hot issue in many industries. That relies on the use of safety critical software in controlling and monitoring critical devices. SCSSs must be certified before installation and operation phase. Certification ensures that the system will not fail consequently will not cause harm to human beings or environment and if it fails, it will fail safely. Considering existing models for developing SCSSs, most of existing methodology did not follow standards, guidelines and related documents for developing SCSSs. The methodology consists of three phases: *Phase1*, safety planning and requirements that consists of four processes (describe the critical system, identify critical system functions, determine the SCSSs safety plan, identify the functions of SCSSs). *Phase2*, safety analysis which consists of (analyze and identify the hazards, apply risk management process, specify safety requirements of critical software systems). *Phase3*, design, implementation, and operation that consists of (design and implement SCSSs, verify and validate SCSSs, certify SCSSs, operate, and maintain SCSSs). This paper is going to focus on safety planning phase.

Literatures showed that safety issue should be considered in the whole critical software systems and from the beginning. Already there

are three methodologies for modeling software safety in safety critical computing systems. Some of them share common processes and differ in other processes. Also the implementation and the sequence of such common processes are different. Although of already exciting models follow standards in developing some process and neglect standards in developing other processes, beside they did not based on standards from early process. The first methodology [1] is a primitive one, it based on four essential processes for developing safety software. These processes are software safety planning, safety critical computer system function identification, software and computing system hazard analyses, and finally validation and verification. The second methodology [2] is a more complex model with more seven processes than the first one. The new processes are software safety requirements analysis, software safety architecture design analysis, software safety detailed design analysis, software safety code analysis, software safety test analysis, software safety evaluation, and software safety process review and documentation. The third methodology [3] nearly is the same process as the second methodology. These models neglecting describing the critical system in which the software is subcomponent of the whole system. They are missing the importance of developing critical software systems according to standards and guidelines. They did not give attention to the process of software certification. The missing processes could produce deficient critical software systems which might cause failures during the operation. For these reasons the proposed methodology for developing SCSSs based on standards and guidelines is presented to overcome the weak points of exciting models.

The paper is organized as follows: section 2 describes SCSSs. Section 3 represents the critical software systems certification. Section 4 presents the methodology for SCSSs safety planning for developing certified SCSSs based on standards and guidelines. Section 5 presents a case study performed according to our methodology. The last section concludes the discussion, and explores trends for future research work.

2 Safety Critical Software Systems

SCSSs can be defined as software that monitors, exercises direct command and controls over the condition or state of hardware components. And if not performed, performed out-of-sequence, or performed incorrectly could result in improper control functions, which could cause a hazard or allow a hazardous condition to exist. SCSSs require rigorous work related to safety analysis, testing and verification to assure safety in the overall system. Software systems are considered as safety critical if it perform one of the following functions [4]:

1) Implement a critical decision making process.
2) Control or monitor safety critical functions of software or hardware.
3) Cause or contribute to hazards.
4) Intervene when an unsafe condition is present or imminent.
5) Execute on the same target system as safety critical software.
6) Mitigate damage if risk occurs.

3 Critical Software Systems Certification

Certification can be defined as "the process of issuing a certificate to indicate conformance with a standard, a set of guidelines, or some similar document." [5]. According to software engineering, certification is typically associated with three meanings: certifying product, process, or personnel. Product and process certification are the most challenging in developing software for safety critical systems such as NPPs, radiation therapy, medical devices, flight control, etc [6]. These critical systems may cause significant damage or loss of life, if not operating properly.

Many governments and international agencies have issued a number of standards, guidelines, and reports related to certification and/or other aspects of software assurance, such as licensing, qualification, or validation, in their specific areas of interest. The goal of certification is to ensure that the safety of critical software systems is satisfied in overall the system and to ensure also the correctness of the system and

ISBN: 978-960-474-304-9

that system is developed according to standards and guidelines [7, 8].

4 A Methodology for Safety Critical Software Systems Planning and Requirements

The proposed methodology describes the development of certified SCSSs based on standards and guidelines. The methodology consists of three phases (safety planning and requirements phase, analysis phase, and design, implementation, and operation phase) as shown in Figure 1. The safety planning and requirements phase consists of four processes as shown in Figure 2. It starts from description of the critical system and its components, relation between components, identification of the critical system functions, specification of SCSSs safety plan, and finally identification of the SCSSs functions.

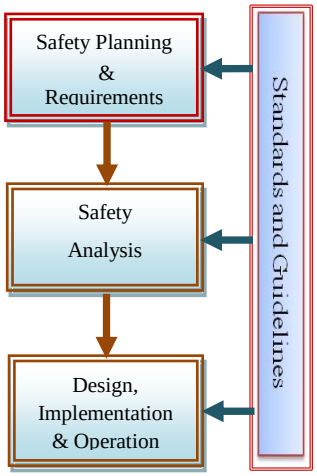

Fig.1 SCSSs Development Methodology

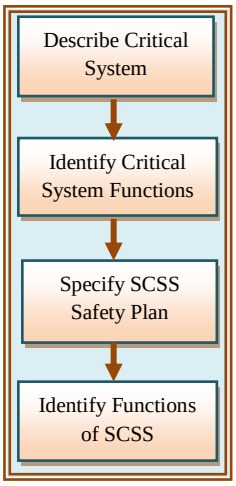

Fig.2 Processes of Safety Planning and Requirements Phase

4.1 Describe the Critical System

The first step is to understand the critical system and its operation, including the resources to be protected. The critical system components as shown in figure 3 include hardware (computer, mechanical electronic), software which monitors or controls the critical system, humans interacting with the system, and system environment. Even that we concerning about the safety of critical software systems but in the same time we cannot neglect to describe and identify all system components because they have affective relation with software system as shown in Figure 3 and in turn affect in the overall safety of the system. The boundaries of that system must be understood, including the interfaces between subsystems and external entities. The critical system must be described in terms of subsystems, components, interface between components, equipments, tools, resources, human factors, and the hazards.

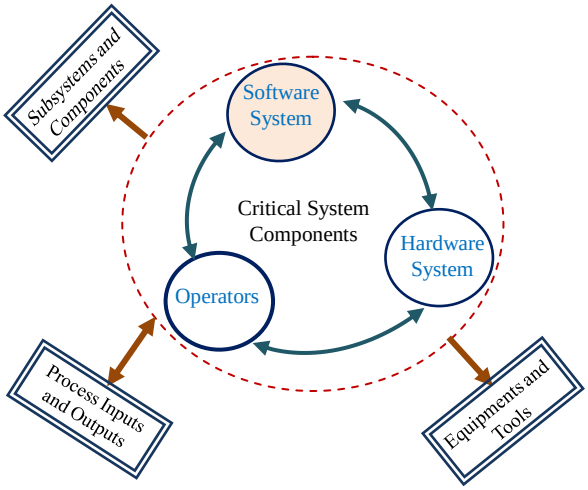

Fig. 3 Critical System Environment

Regarding to the effective relation between SCSS and other components in the critical system and according to NASA-STD-8719.13A [9], software can be used to detect and control hazards of critical system components (hardware, and operator), but software failures can also contribute to the occurrence of hazards. Some software hazard causes can be eliminated with hardware controls. For every hazard cause, there must be at least one control method, usually a design feature (hardware and/or software). Examples of hazard causes and controls are given in Table 1.

4.2 Identify Critical System Functions

The requirements analysis and functions identification should be extended to critical system with all components not limited to software. Because software is a single component of the system.

According to IEC 61508 [10] critical system functions should include the following:

ISBN: 978-960-474-304-9

1) Function descriptions, indicating the intent of the functions.
2) Operation descriptions, indicating what the system is supposed to do in all major phases of operation.
3) Subsystems and components that make up the critical system.
4) Process inputs and outputs of the critical system.
5) System data.
6) Critical system equipments and tools.

Table 1 Hazard Causes and Controls

Cause	Control	Example of Control Action
Hardware	Software	Fault detection and saving function, or event checks which activate or prevent hazardous conditions.
Software	Hardware	Hardwired timer or discrete hardware logic to screen invalid commands or data. Sensor directly triggering a safety switch to override a software control system. Hard stops for some events.
Software	Software	Two independent processors, one checking the other and intervening if a fault is detected. Emulating expected performance and detecting deviations.
Operator	Software	Software validation of operator-initiated hazardous command. Software prevents operation in unsafe mode.
Software	Operator	Operator sees control parameter violation on display and terminates process.

4.3 Specify Safety Critical Software Systems Safety Plan

Software safety planning is performed in the context of software safety management which includes responsibilities of applying the defined safety principles, criteria, safety targets, purpose, objectives of software safety program, generic safety requirements, and design tools approved for use in the system. Which aid in achieving and establishing safety plan to meet specified requirements. The software management plan is formatted, and documented in accordance with NASA-STD 2100-91 [11], section 306 of NHB 1700.1 (V1-B) [12], and IEEE STD-1228-1994 [13]. The plan should describe how the activities specified by this standard will be implemented. The plan should specify the activities to be carried out, the schedule on which they will be implemented, and the products that will result. The plan is addresses the interrelationships among system safety analysis, software safety analysis, and the software development efforts. The plan is specifically addresses the mechanism by which safety critical requirements are generated, implemented, and verified. Additional information about software safety planning can be found in [2].

4.4 Identify the Functions of Safety Critical Software Systems

Once safety critical system functions had been identified, the functions of safety critical software system will be easily identified [4] such as:

1) Controlling, or monitoring safety critical functions of software or hardware.
2) Intervenes when an unsafe condition is present.
3) Handle safety critical data either providing information related to safety or analysis critical data.

Identifying these functions help in identifying the special measures that should be required for understanding and mitigating risks.

At this step of the proposed procedure, the top level and general requirements are defined [1]. These requirements are general not related to specific hazard but derived from safety critical functions knowledge, safety standards, design standards, mishap reports, experience and lessons gained from similar software [2]. This process compliances with standards MOD0055'89 [14], ISO 26262 [15], and JSSC Software System Safety Handbook [16].

5 Case study: An Insulin Pump Control System

The insulin pump is a medical system that mimics the operation of the pancreas. It used by people who cannot make their insulin and suffering from diabetics. It helps to achieve rapid, precise, and control varying glycemic. The critical insulin delivery system is used to monitor the blood sugar levels and deliver an appropriate dose of insulin when required. Modern insulin pumps depend increasingly on embedded control software. Software is responsible for safety functions such as sensing

ISBN: 978-960-474-304-9

blood data, providing display output, dosage control, and mitigating certain hazards through alarms and alerts. To develop insulin pump system with high assurance of its reliability and safety, we should follow IEC 61508 [10]. The insulin pump system includes eleven components (pump, needle assembly, insulin reservoir (inside the pump), data stores, blood sugar sensor, power supply, controller, communication, displays, alarms, and clock components) as shown in Figure 4. The entire system is controlled by the controller component. The measuring time and compared blood sugar level data are stored in data stores component. These data are used to analyze the diabetic's health condition. The insulin is stored in the reservoir component and pumped from the reservoir to needle assembly by using pump component. The insulin devise is communicated to doctor's computer through communication component to download system setup parameters and upload the data stored. Alarm component using to alert the user and set status to warning.

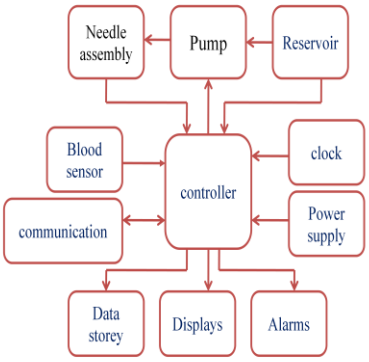

Fig.4 Insulin Pump System Structure

The insulin pump safety plans should include 1) Functional description. 2) Data resources which show all data items, and their relations to different operations. 3) The constraints which indicate the restrictions on the functionalities of operations or data. 4) Hazard analysis by using Fault Tree Analysis (FTA) to identify the hazard causes, and apply risk management techniques. 5) Safety requirements identification. 6) Safety validation and safety assurance by using formal methods and safety proofs. 7) Safe design principles applied. 8) Verification activities and methods including (software analysis, software inspection, and software test). 9) System testing for both normal and abnormal operation including (software/hardware integration test, interface test, and operation test). The function of insulin pump system is to monitor the blood sugar levels and deliver an appropriate dose of insulin when required. This function requires reading the blood parameters using sensor, then analyzing the blood parameters, and monitoring the blood sugar levels. Consecutive readings are compared and if they indicate that the level of glucose is rising, compute the required insulin dose then actuate insulin pump signal to inject the insulin to counteract this rise. The current sugar level and required dose are displayed to the user. Also the system issuing alerts to the user in case of system sugar level is low. The insulin dose computed according to sugar levels (unsafe, safe, and undesirable). The computed dose of insulin injection depends on cumulative reading at interval times not on absolute level of glucose. There are four scenarios for injection 1) level of sugar in safe band. 2) Level of sugar is failing. 3) Level of sugar is stable. 4) Level of sugar is increasing. The embedded critical control software is responsible for reading blood parameters, measuring level of blood sugar at periodic intervals, comparing between the consecutive readings, computing required insulin dose, actuating insulin pump signal, issuing alerts to the user, coordinate the functions of the various components within the pump to ensure safe operation of the pump.

6 Conclusion

In this paper a new methodology for developing certified SCSSs plans is presented. This methodology which is commonly used for developing certified SCSSs consisting of three phases. This paper discussed deeply the first phase and related processes. These processes inspired on many standards, guidelines, and related documents for developing software systems generally and SCSSs specifically. This phase of methodology relies on a generic and domain specific standards. The proposed safety and requirements phase of methodology integrates and complements these standards.

ISBN: 978-960-474-304-9

This phase of methodology is applied to the insulin pump system. The main advantage and contribution of this methodology is that the proposed methodology help the practitioners to developed certified SCSSs beside other advantages such as overcoming the weak points of the existing methodologies for developing SCSSs. This paper represents a milestone for developing certified critical software. It shed the light on the importance of embedding standards and guidelines in the analysis, design, implementation and testing processes of SCSSs. Our future work is dedicated for completing the methodology. It will discuss in detailed phase 2 and phase 3. And apply this methodology for developing NPPs system.

References:

[1] Daniel P. Murray, Terry L. Hardy, ‘‘Development Safety Critical Software Requirements for Commercial Reusable Launch Vehicles’’, National Technical Information Science, 2009.

[2] S.Phani Kumar, P. Seetha Ramaiah, V. Khanaa, ‘‘Identification of Software Safety Metrics for Building Safer Software based Critical Computing Systems’’, International Journal of Computer Science and Application, ISSN 0974-0767, 2010.

[3] Srinivas Acharyulu, P.V., P. seethe ramaiah, ‘‘A Methodological Framework for Software Safety in Safety Critical Computer Systems’’, Journal of Computer Science 8 (9), ISSN 1549-3636, Science Publications, 2012, pp. 1564-1575.

[4] NASA STD-8719.13. NASA software safety guidebook: NASA technical standard. Department of Defense, 2004.

[5] Neil Storey, Safety-Critical Computer Systems. Addison-Wesley, 1996.

[6] Andrew Kornecki, Janusz Zalewski, “Software Certification for Safety-Critical Systems: A Status Report”, Proceedings of the International Multi conference on Computer Science and Information Technology, IEEE, 2008, pp. 665 – 672.

[7] NASA/CR–2003-212806. Certification Processes for Safety-Critical and Mission-Critical Aerospace Software. Stacy Nelson. June 2003.

[8] Dr. Holger Giese. Slides of lecture Software Engineering for Safety Critical Computer Systems, Software Engineering Group, University of Paderborn, 2003.

[9] NASA Software Safety Guidebook, NASA Technical Standard, NASA-GB-8719.13A, 2004.

[10] IEC 61508, International Standard, Functional Safety of Electrical /Electronic /Programmable Electronic Safety- Related Systems, 1998.

[11] NASA-STD-2100-91, NASA Software Documentation Standard Software Engineering Program, 1991.

[12] NHB 1700.1(V1-B), NASA Safety Policy and Requirements Document, 1993.

[13] IEEE STD 1228-1994. IEEE Standard for Software Safety Plans. 1994.

[14] MOD0056'89, Interim Defense Standard 00-56, (DRAFT) Requirements for the Analysis of Safety Critical Hazards, Ministry of Defense, UK, May 1989.

[15] International Organisation for Standardiz-ation. ISO 26262: Road Vehicles-Functional Safety, Draft International Standard (DIS), 2009.

[16] D. Alberico, J. Bozarth, M. Brown, et. Al. JSSC Software System Safety Handbook; A Technical and Managerial Team Approach December 1999.