Journal of Multimedia and Information System Vol. 2, No 2, June 2015(pp. 215-220): ISSN 2383-7632(Online)
http://dx.doi.org/10.9717/JMIS.2015.2.2.215

A Study on the Software Safety Assessment of Healthcare Systems

Author: Rafal Olenski[1,*] and Man-Gon Park[2]

Abstract

The safety-critical software in healthcare systems needs more and more perceptive excess among human observation and computer support. It is a challenging conversion that we are fronting in confirming security in healthcare systems. Held in the center are the patients—the most important receivers of care. Patient injuries and fatalities connected to health information technologies commonly show up in the news, contrasted with tales of how health experts are being provided financial motivation to approve the products that may be generating damage. Those events are unbelievable and terrifying, however they emphasize on a crucial issue and understanding that we have to be more careful for the safety and protection of our patients.

Key Words: Safety-Critical Software Components, Software Validation Guidelines ISO 14971, IEC 62304, Fault Tree Analysis Method for Safety-Critical Software Components.

I. INTRODUCTION

The importance of software is starting to be progressively significant, and it is in use in many critical applications, such as avionics, transportation control systems, health systems (which we will focus on in this paper), engineering, power systems, and sensor networks [1]. As we know well, safety-critical systems can cause mishaps and hazards. Software become dangerous if it can follow to a threat i.e. cause other mechanisms to be harmful or if it is controlling the hazard. Software is thought of as harmless if it is not possible or doubtful that the software might ever develop activities that would follow a tragic result for the system that the software is responsible for. Cases of catastrophic activities contain loss of physical property, physical injury, or death. Software engineering of a safety-critical system involves a perfect understanding of the software's part, and collaboration together with the system [2, 3]. All systems need the maximum care in their design, specification, application, process and conservation, as they might lead to damages or death, and also as an effect in material loss.
IT technology is used in a medicine more often than people think. A microprocessors are used to control an insulin pump is well known. The fact that a pacemaker is mostly a computer is less recognized. Widespread use of information technology in surgical actions is mysterious for ordinary people. Modern tools are making innovations in techniques such as spinal surgery, hip replacement and many other surgical procedures. In those above cases, computer controlled robotic devices are changing the surgeons old-style instruments, and providing significant profits to patients [4].
In this paper, we discuss about the software safety assessment to classify and mitigate the risks related to malfunctioning software in the medical devices of healthcare systems.

II. SAFET-CRITICAL SOFTWARE COMPONENTS IN HEALTHCARE SYSTEMS

Software-based medical devices became a serious division of the healthcare scene. Various health devices need to interact together with gear, associate with clinic and laboratory information systems, and work in extreme circumstances. The improved expectations on such devices and their rising universality have created

Manuscript received June 18, 2015 ; Revised July 20, 2015 ; Accepted August 10, 2015. (ID No. JMIS-2005-15)
Corresponding Author(*): Rafal Olenski, Department of IT Convergence and Application Engineering, Pukyong National University, Korea, +82-51-629-6245, rafalolenski@gmail.com.
[1]Department of IT Convergence and Application Engineering, Pukyong National University, Korea, rafalolenski@gmail.com
[2]Department of IT Convergence and Application Engineering, Pukyong National University, Korea, mpark@pknu.ac.kr

challenging design tasks for their creators. The most important is to confirm safety. It has become more demanding because of the instant growing complexity of surrounded software. For the reason that software engineering is a fundamental human process, it is not likely or even impossible to create software without errors. An important task for device developers is to recognize and mitigate the hazards related with surrounding malfunctioning software in devices. Health devices integrate many types of features. For example, malfunctioning electric modules, a defective software component will have dramatic results. Though, other different kinds of modules, classifying and calculating the possible effects of malfunctioning software mechanisms are additionally problematic. Because of the growth of complexity, it results in an additional amount of weaknesses. Another reason is that many devices share similar mechanisms, such as controllers and pumps, those elements have created path record. Engineers usually deliver device developers with performance files for these common elements. In contrast of software, it is often patented and established by medical device developers for a purpose in an exact device. Unfortunately, there is no well-known path record for software components. Therefore, the responsibility depends on a device developers to guarantee that software-based medical devices are harmless and efficient. To resolve such a challenge, it demands knowledge in efficient risk management work, understanding the software safety, and the implementation of a risk management outlook.

Regardless of their significant benefits, software-based services and systems can pose hazards to patient health [5, 6]. For instance, between 2005 and 2009, the US Food and Drug Administration got over 56,000 reports of issues linked to the use of infusion pumps [7]. Many of the safety matters were marked out as software defects. In the United Kingdom, the Medicines and Healthcare Products Regulatory Agency informed a constant growth in medical device adverse incidents, 9099 reports in 2009 [8]. The British Medical Journal also stated an important growth in medical device recalls and warnings [9]. Specified by the criticality of certain software systems, e.g. EHR, measuring the level of which the software actions contribute to safety hazards in healthcare services must be an fundamental part of the medical threat valuation process and the general clinical safety matter [10].

These safety dangers rise in medical environments that are centered on the connections among many different human, technical and high-tech elements. Understanding and adjusting the complex links between the software's behavior and the emergence of the medical hazards (i.e. possible to cause escapable/unintentional harm) is a great challenge. Talking about this challenge at the medical level needs close relationship among different investors, mainly clinical authorities, health experts, safety analysts and systems and software engineers [11].

Device producers are ethically, legally, and financially responsible to guarantee that their development creations do no damage. However, in spite of the huge amount producers invest in authorizing the security of their products, catastrophes continue to occur. For instance, the Food and Drug Administration informed that: among 1990 and 2000 there were 200,000 pacemaker recalls because of the software issues. In the U.S, from 1985 to 2005, there were 30,000 fatalities accidents and 600,000 damages caused by medical devices, 8% involved a defective software [12].

Fig. 1. Infusion Pump

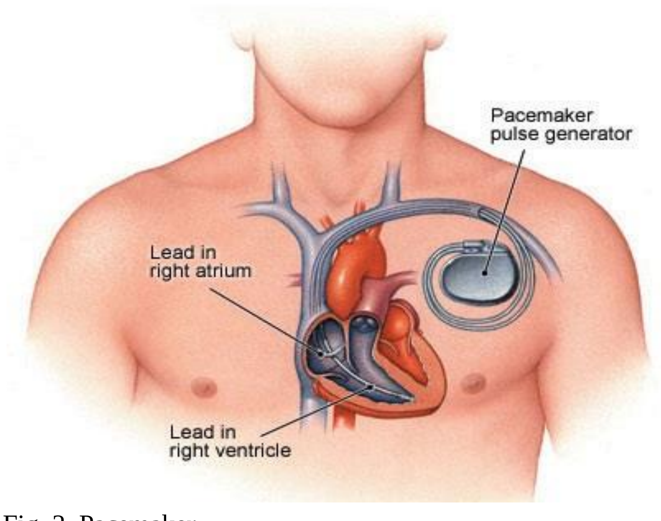

Fig. 2. Pacemaker

Fig. 3. Insulin Pump [21]

III. SOFTWARE VALIDATION GUIDELINE FOR HEALTHCARE SYSTEMS

The complication of the software implementation in many clinical devices means that confirming their safety needs complete testing with a compounding of other methods such as design validation, implementation validation, and remaining fault assessment. Failures in medical devices don't usually mark the headlines the way airplane or train mishaps do. For patients, most likely, medical device errors can have catastrophic results [13].

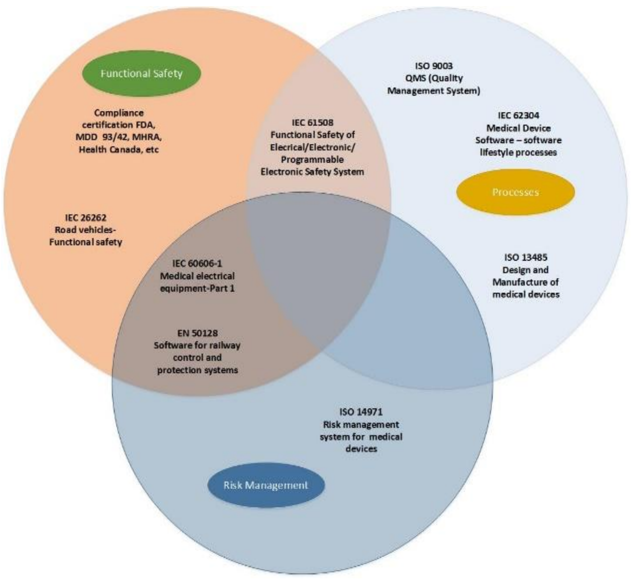
Functional Safety
Compliance certification FDA, MDD 93/42, MHRA, Health Canada, etc
IEC 61508 Functional Safety of Elecrical/Electronic/ Programmable Electronic Safety System
ISO 9003 QMS (Quality Management System)
IEC 62304 Medical Device Software – software lifestyle processes
Processes
IEC 26262 Road vehicles- Functional safety
IEC 60606-1 Medical electrical equipment-Part 1
ISO 13485 Design and Manufacture of medical devices
EN 50128 Software for railway control and protection systems
ISO 14971 Risk management system for medical devices
Risk Management

Fig. 4. Functional Safety Related Standards in Medical Devices [15].

International Organization for Standardization is an international federation of nationwide standards bodies (ISO member groups). The work of making International Standards is normally approved by ISO technical committees. Each associate group involved in a subject for which a technical committee has been established, has the right to be represented on that committee. International organizations, governmental and non-governmental, in collaboration with ISO, also take part in the work. ISO cooperates diligently with the International Electro Technical Commission (IEC) on all issues of electro technical standardization to have medical systems protected. [14]

Focusing on IEC 62304, which is a worldwide standard for medical device software life cycle development, it isn't connected to functional safety. As an alternative, it reports the "framework of life cycle processes with activities and tasks necessary for the safe design and maintenance of medical device software" and, according to ISO 14971, the risk management associated with those processes [16].

For the reason that IEC 62304 is not about functional safety, it doesn't describe acceptable failure rates in numbers. Compliance with standards of IEC 62304 doesn't suggest a safety integrity level (SIL) as does, for instance, conformity to IEC 61508, which is worthless without one and others. Even though IEC 62304 sets out the procedures essential to create an efficient device, it is not well known how the assessment of those procedures is linked to the value of the device manufactured.

Act in accordance with the development processes was defined in IEC 62304 we have to perform the essential examination to guarantee that the new invention is safe. First of all, engineers must start with the principle that all software has errors, and these mistakes may lead to disasters. Damages are the consequence of multiply situations that begin with a wrong introduced into a design or application. Faults may lead to errors, and errors may lead to failures [15].

An additional well-known issue is to postpone risk management till device designers have finished the design- method that minimizes risk mitigation opportunities. ISO 14971 states that, when developers try to minimize risks, they must follow three design rules as a priority. Modify the design to remove threats- If applying the first rule is impossible, follow safety measures in the device or manufacturing route, containing the skill to identify circumstances that might follow to the threat's happening. If an adoption or implementation of those two rules is difficult, adding documentation in an operator's guidebook to clarify precautions to take if circumstances that might lead to a hazard to occur. [14]

Certainly, these values highlight an initial beginning to risk management and therefore it gives more chances and freedom for the device creator in decreasing threat in the same time with the device progress [7].

ISO 14971 explains a risk as a possible source of damage—physical harm or damage to the health of people (patients, clinicians, and third parties), property, or the environment. The basic requirements device developers are to classify all known and predictable threats and measure each hazard's severity—the amount of its probable effects. Common threats for exact devices are a valuable beginning. According to ISO 14971 we can classify hazards that creators never assumed to happen. The important standard is that if a threat can actually occur, be sure that it will. However, we will not concentrate on the hazard's probability of occurrence but on the damage that may result [17].

Systems always include: hardware, the software, the users, and the surroundings. Everything needs to be thought out well during the developing of the software. Altogether fragments of the system need to be harmless. Theoretical or practical security begins at the system level of quality. Security can't be guaranteed if we just concentrated only on software. We can create a software without 'bugs' and implied several security features, however we can't predict how software will act with all components in the system.

The system safety analyses are the initial point to classify software safety requirements essential to help to create the software requirements specification. Such a requests have to be delivered to the developer and attached into the software requirements data.

During the whole project life cycle, the system safety analysis must be performed. The software safety examination procedure must last to evaluate the effects of the systems analyses to declare that modifications and answers at the system level are combined into the software as required. Additionally, the software safety analyses deliver input to the system safety analyses. The software safety analyses are an important part of the complete system safety examination and they cannot be conducted separately. As a result, we have four security-relevant elements of a system development route: 1. Classifying threats and associated safety requirements, 2. Creating the system to face its safety requirements, 3. Examining the system to display that it comes through its safety desires, and 4. Proving the safety of the system by manufacturing a safety case [18].

IV. FAULT TEE ANALYSIS METHOD(FTA) FOR SAFETY CRITICAL SOFTWARE COMPONENTS OF HEALTHCARE SYSTEMS

Device developers have to emphasize on classifying hazards first and then recognizing failure modes that can follow to those dangers. FTA and failure modes effect criticality analysis are one of the best tools. ISO 14971 contains this condition: The producer will guarantee that those conducting risk management assignments contain individuals with knowledge and experience proper to the work given to them.

Fault Tree Analysis (FTA) is a logic diagram showing the routes to an event, it is a procedure to recognize threats, and it is covered by a complete examination to find out what could cause it. The event under the study is called the 'Top Event'. The 'Top Event' causes are diagramed using typical logic gate symbols (AND- the output incident happens when all input events happen at the same time. OR -the output happening occurs when at least one of the input happenings occur).

Fault Tree Analysis usually take five steps. The first is to describe the undesired event to study, the 'Top Event' – states the unwanted happening that can cause risk. The second is to understand the system. We need to describe the events that could let the 'Top Event' happen. For each event express what would cause it. Carry on to analyze the system. The third is a creation of the fault tree– after selecting the undesired happening and analyzed the system to identify the causal events. Define the events and their relationships using AND and OR gates (more complex gates are also possible). As the fourth step, evaluate the fault tree- look for possible improvements that can mitigate, reduce, or eliminate the events. Classify all probable hazards effecting in a direct or indirect way of the system. Lastly, control the hazards identified– after recognizing the events and hazards, determine the methods to decrease the possibility of occurrence.

In the case of software hazards, the common attention is to define faults that will cause the system to fail to deliver a system service, such as a monitoring system. Fault tree is constructed to connect all the possible circumstances together, to help classify the interrelationships of the failures, which modules may cause them, and what result there might be.

Here is an example of a fault tree, as applied to the insulin delivery system, a personal insulin pump for people suffering from diabetes. It is an external device that mimics the function of the pancreas. It uses a fixed sensor to calculate the blood sugar level at periodic intervals and then injects insulin to keep the blood sugar at a 'normal' level:

Notice that this tree is incomplete, since only the possible software faults are illustrated on the figure 2. The probable failures related to hardware, such as blood monitor, low battery, or sensor failure, patient overexertion or carelessness, or medical staff failure are not included in this diagram

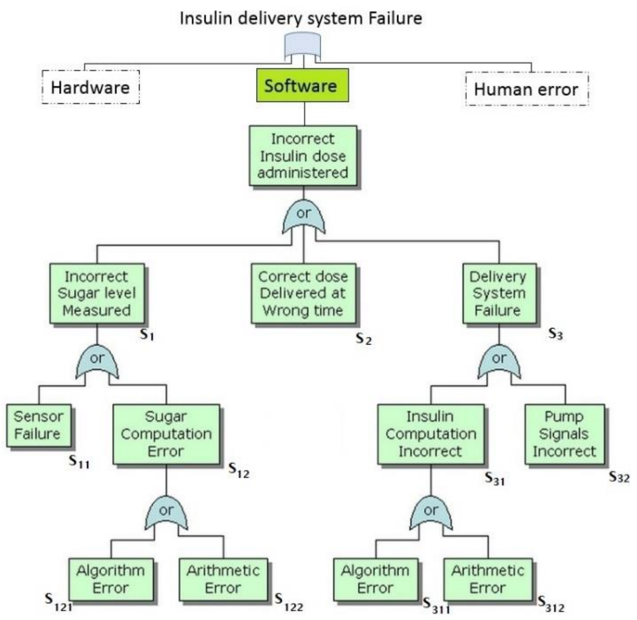

Fig. 5. Fault Tree Diagram of Insulin Delivery System

The fault tree is useful tool to help with system risk assessment tasks. Once the risks are recognised, there are other valuations that need to take place. First, the probability of the risk occurrence must be measured. This is often computable, therefore numbers may be matched based on MTBF (Mean Time between Failures), latency effects, and other well-known objects. There may be other immeasurable contributors to the risk probability, however, such that these must be evaluated and estimated by the specialists in the field. We should never make short this process with critical systems. Lastly, the risk assessment must contain the severity of the risk, an estimation of the cost to the development in the happening the risk item actually does take a place. That means all associated with costs, containing human injury, program delays, corruption to hardware, damage of data, etc.

Pr (Probability of Software Failure in Insulin System)

$= \Pr(S_1) \text{ x } (S_2) \text{ x } (S_3)$

$= [\Pr(S_{11}) \text{ x } (S_{12})] \text{ x } \Pr(S_2) \text{ x } [\Pr(S_{31}) \text{ x } (S_{32})]$

$=[\Pr(S_{11}) \text{ x } (S_{121}) \text{ x } (S_{122})] \text{ x } \Pr(S_2) \text{ x }$

$[\Pr(S_{311}) \text{ x } (S_{312}) \text{ x } (S_{32})]$

$= p_{11} \text{x} p_{121} \text{x} p_{122} \text{x} p_2 \text{x} p_{311} \text{x} p_{312} \text{x} p_{32},$

if we denote that Pr(S*) = p*.

S* (An Event Cause Threats)

S_1(Incorrect Sugar level Measured)

S_2(Correct Dose Delivered at Wrong Time)

S_3(Delivery System Failure)

S_{11}(Sensor Failure)

S_{12}(Sugar Computation Error)

S_{31}(Insulin Computation Incorrect)

S_{32}(Pump Signals Incorrect)

$S_{121,311}$(Algorithm Error) Compare dose to be delivered with previous dose or safe maximum doses. Reduce dose if too high.

$S_{122,312}$(Arithmetic Error) A computation causes the value of a variable to overflow or underflow. Maybe include an exception handler for each type of arithmetic error.

VI. CONCLUSIONS

The medical device software development area is full of procedures that software development organizations need to conform with in order to market their products. In this paper we have described these adjusting standards. This paper is proposing one of the methods: Fault Tree Analysis and discussed the principles relevant to software safety. We focused on the standard for medical device risk management ISO 14971:2007 which is recognize as a compatible standard by FDA (Food and Drug Administration). The European Union lists it as a consistent standard to the MDD (Medical Device Directive), IVD (In Vitro Diagnostics), and AIMD (Active Implantable Medical Device). ISO 14971 fits perfectly for the risk management. No matter the marketing region (US, Canada, UE, etc.) is of valuable addition to medical devices QMS (Quality Management System), it is the most effective when it is integrated into companies QMS. Furthermore, it will be essential to adopt risk management from the initial stage until the product is done rather than having it as an afterthought. Avoiding this could hinder the development process as security or risk errors detected later will require re-coding and analysis. Software development teams need to practice secure software development life cycle in their products to promote software safety.

Careful consideration of the above features and practices will lead to the reduction of hazards of software defects.

REFERENCES

[1] Committee on Patient Safety and Health Information Technology of the Institute of Medicine, *Health IT and Patient Safety: Building Safer Systems for Better Care*, National Academies Press, Washington D.C., 2011.

[2] D. Wang, F. B. Bastani, and I. L. Yen, “Automated Aspect-Oriented Decomposition of Process-Control

Systems for Ultra-High Dependability Assurance," *IEEE Transactions On Software Engineering*, Vol. 31, No. 9, pp. 713-732, September 2005.

[3] P. V. Bhansali, "Software Safety: Current Status and Future Direction," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 1, p.3, January 2005.

[4] R. R. Lutz, "Software Engineering for Safety: a Roadmap," *Proceedings of the Conference on the Future of Software Engineering*, pp. 213-226, 2000.

[5] J. C. Knight, "Safety Critical Systems: Challenges and Directions," *Proceedings of the 24th International Conference on Software Engineering (ICSE)*, pp. 547-550, 2002.

[6] R. H. Taylor and D. Stoianovici, "Medical Robotics in Computer-Integrated Surgery," *IEEE Transaction on Robotics and Automation*, Vol. 19, No. 5, October 2003.

[7] R. Rakitin, "Coping with defective software in medical devices," *IEEE Computer, Vol. 39, No.* 4, pp. 40-45, 2006.

[8] R. Koppel, J. P. Metlay, A. Cohen, B. Abaluck, A. R. Localio, S. E. Kimmel, and B. L. Strom, "Role of computerized physician order entry systems in facilitating medication errors," *The Journal of Urology*, 2005.

[9] U.S. Department of Health and Human Services Food and Drug Administration, "Infusion Pump Premarket Notification", Total Product Life Cycle, 2010.

[10] I. Habli, A. Al-Humam, T. Kelly, and L. Fahel, Medical and Health Care products Regulatory Agency, Adverse Incident Reports, 2009.

[11] C. Heneghan, M. Thompson, M. Billingsley, "Medical device recalls in the UK and the device regulation process: retrospective review of safety notices and alerts," BMJ Open 1(1): e000155, May 2011, http://dx.doi.org/10.1136/bmjopen-2011-000155.

[12] Health and Social Care Information Centre, "Clinical Risk Management: it's Application in the Manufacture of Health IT Systems," Report-ISB 0129, 2013.

[13] D. Jackson, "Software for Dependable Systems: Sufficient Evidence?" Washington, DC: National Academies Press, p.23, 2007.

[14] ISO 14971:2007, *Medical Devices - Application of Risk Management to Medical Devices,* ISO Standard Catalogue, Reviewed on 2010.

[15] C. Hobbs, *Build and Validate Safety in Medical Device Software,* Medical Electronics Design, January 2012.

[16] IEC62304:2006, "Medical Device Software-Software Lifecycle Processes," Geneva: International Electro-technical Commission, 2006.

[17] M. Thomas, "Engineering Judgment," Australia: Australian Safety Critical Systems Association, 2004.

[18] J. McDermid, "Software Hazard and Safety Analysis," *Formal Techniques in Real Time and Fault Tolerant systems*, Lecture Notes in Computer Science, Vol.2469, pp. 23-34, 2002.

AUTHORS

Rafal Olenski graduated with the B.A. and M. A. in Physical Education at the Gdansk University of Physical Education and Sport, Poland in 2010. He is a research member of the Software Engineering and Multimedia Information Systems Lab. as well as a Ph. D. student of the Dept. of IT Convergence and Application Engineering, Graduate School, Pukyong National University, Rep. of Korea. His research interests are in Security Analysis, Safety Analysis, and Big Data Analysis Methods for Healthcare Systems.

Man-Gon Park is a head professor of the Dept. of IT Convergence and Application Engineering, College of Engineering, Pukyong National University, Republic of Korea since 1981. Also he was the president and chairman of the Korea Multimedia Society (KMMS). He served as the Director General and CEO of the Colombo Plan Staff College for Technician Education (CPSC) from 2002 to 2007, which is an intergovernmental international organization of 29 member governments for Human Resources Development in Asia and the Pacific Region. He has been the visiting professor at the Department of Computer Science, University of Liverpool, UK; exchange professor at the Department of Electrical and Computer Engineering, University of Kansas, USA; and visiting scholar at the School of Computers and information science, University of South Australia; visiting professor at the Department of Computer Science and Engineering, University of Colorado, Denver, USA.

He was dispatched to Mongolia and People's Rep. of China by KOICA on various projects as information systems consultant. He has also embarked on consulting works and conducted training programs in ICT on individual capacity for Korean groups of companies, governmental and non-governmental agencies and other institutions in Korea. His main areas of research are software reliability engineering, software safety and security engineering, business process reengineering, Internet and web technology, multimedia information processing technology, and ICT-based human resources development