Hazard Analysis and Metrics Identification for Software Safety in Medical Cyber-Physical Systems

Kadupukota Satish Kumar and Panchumarthy Seetha Ramaiah
Dept of Computer Science & Engineering, JNTU Kakinada, India

Abstract
The safety of software is becoming progressively important in computer controlled systems on which human life depends. In many cases software perform a main role in the safety-critical systems. There are many well known examples in application areas such as medical devices, aircraft flight control, weapons, and nuclear systems. In the field of medical sciences, innovative implantable medical devices are increasingly managed by software. In the medical domain, cochlear implant system (CIS) for bionic ear is one of the safety-critical medical devices which is controlled through clinical programming soft-ware (CPS). This paper examines the nature of software related failures of medical devices and applied this analysis to CPS of CIS as Medical Cyber-Physical System (MCPS) of bionic ear and performs the hazard identification, hazard analysis, safety design. The process begins with different types of hazard analysis techniques to CPS such as: FMEA, FTA, PHA, ETA, and FC. All the techniques are applied and then a comparison is made among them and results are explained.

INTRODUCTION
Safety-critical systems (SCSs) are those systems whose failure could result in loss of life, significant property damage, or damage to the environment [11]. Safety is involved with security of human life, the environment and property. The purpose of a software safety analysis is to identify hazards, demonstrate the absence of specific hazards, and determine the possible damaging effects resulting from hazards, determine the causes of a hazard, identify safety design criteria that will eliminate, reduce, or control identified hazards, evaluate the adequacy of hazard controls and prevent the execution of a safety critical function. A software failure must be a design or implementation error which is not detected during the testing phases. Software errors in medical devices are execution failure, data failures, and communication failures. Execution failures can be caused by an erroneous call, jump, and task control process. Data failures are caused by incorrect variable usage, undetected support software design errors, erroneous data access procedures. Communication errors are caused by misunderstanding of software interface specifications. Examples for safety critical medical devices are radiation therapy machines, medical monitoring, and medical robots [7].

A software bug can be defined as that part of the code which would result in an error; fault or malfunctioning of the program [17]. Some bugs can be detected easily during development. But some bugs will be found late in the development process. These are low probability errors which are hard to detect and occur on very sparse set of inputs [17]. According to IEEE standards, a bug is an incorrect instruction in a program. A failing is triggered because of a bug and may alter the exterior actions of the system. The major categories are requirement and functionality bugs, structural bugs, data bugs, coding bugs, interface, integration and system bugs, test and test design bugs. These bugs increase the cost and development of CPS of cochlear implant system. The primary task of programmer is minimizing these defects, identify and eliminate existing bugs early in the development process. The defects detected at the beginning of development will cause much smaller harm than those which are recognized later in the utilization of the application. A common category of bugs can be defined in accordance with the regularity of the incident of that bug and severity of that bug. The effect of bug depends on the software and the system. Some of the bugs can have catastrophic consequences. Any consequence that results in either death, injury, damage to property or damage to the environment is called mishap or accident from the point of view of system safety standard [14]. A state of the system that, possibly in combination with environmental conditions, leads to a mishap is called hazard [14]. An internal or external condition (or combination with both) leads to a hazard is called hard cause [14]. The identification of hazards (unsafe states), hazard causes and measures that can be taken to eliminate or control the hazard or to reduce the risk is called hazard analysis [14]. An assessment of the consequences of the worst possible mishap that could be caused by a specific hazard is called hazard severity [14]. How-ever we can generally have following sessions for severity: Catastrophic: potential of multiple deaths or serious injuries. Critical: Defects that could cause serious consequences for the system like losing some important data or potential of death. Marginal: potential of injury. Negligible: These may not necessarily hamper the system performance, but they may give slightly different interpretation and generally avoidable. Software in medical devices plays an increasingly important role in health care [12]. Defibrillator’s, dialysis machines, surgical devices, pacemakers are the examples of different safety critical systems.

The remainder of this paper is organized as follows. Section 2 describes the safety analysis in medical devices. Section 3 describes the designing safety. Section 4 presents the application of software safety as CPS. Section 5 discusses the functional operation of cps and properties. Finally, Section 6 gives conclusions.
SAFETY ANALYSIS IN MEDICAL DEVICES

The role of software has become increasingly important and is being used in many critical applications, such as avionics, vehicle control systems, medical systems, manufacturing, power systems, and sensor networks [18], [4]. With the increased usage of software there is a need for a system who’s failure is not attributed to software. For this we need to do a software safety analysis. Software safety analysis is the process of collecting and analyzing data to determine the cause of a failure and how to prevent it from recurring [8]. Safety analysis may be conducted on the design stage and on field use stage of the product life cycle.

In the field of medical sciences, some of the devices had failed erroneously, causing some damage, for which software has been identified as the main culprit. Following are some of the such accidents:

- A programmable heart pacemaker suddenly froze while it was being adjusted by a doctor. A device of dispensing insulin delivered the drug at an inappropriate rate resulting in a patient receiving a drug overdose [15].
- An ultrasound scanner sometimes underestimated fetal weight [15]
- A blood analyzer displayed incorrect values because addition, rather than subtraction had been programmed into a calibration formula. [15]
- A bug in the code controlling the Therac-25 radiation therapy machine was directly responsible for at least five patient deaths in the 1980s when it administered excessive quantities of X-rays.
- A Medtronic heart device was found vulnerable to remote attacks in March 2008.
- Tens of thousands of medical devices were recalled in March of 2007 to correct a software bug. According to news reports, the software would not reliably indicate when available power to the device was too low.
- In one study of 500 implant recipients at one implant center between 1989 and 2006, 51 out of the 500 (10.2%) had to have revision surgery.
- In another study of 720 patients in South Korea between 1990 and 2007, 30 (4.2%) had to have revision surgery. Of these, 12 were reimplanted.

Hazard, Causes, Controls and Metrics in MCPS

A hazard is any potential condition that can cause: damage, lack of life to personnel; harm to or loss of a system, devices, or property; or harm to the surroundings. The properties of hazards are cause, control, and verifications. Causes are the root reason for the incident of a dangerous condition. Control is an attribute of the design of hardware or software that avoids a hazard or reduces the hazard. Verification is a method for assuring that the hazard control has been implemented and is sufficient through test, analysis, inspection, simulation or demonstration. Each hazard (e.g., Erroneous release of volume too high) has one or more causes. Each cause has one or more controls (e.g., Placement of warning labels, warning lights and buzzers are used for risk mitigation) and each control has one or more verifications to ensure that the control is appropriately implemented.

Software metrics are well-known indicators for software quality analysis [10], [11]. Software metrics are used to measure the software quality of system. Software metrics can be divided according to their usage. Some metrics are software in hazards, causes, and controls metrics and some of them are program metrics used for measuring characteristics of system (i.e. number of lines of code, number of methods etc.). The number of software-related causes and controls of hazards are used to quantify the significance of software. Software safety metrics are used to provide quality assurance on the state of software safety.

Some of the metrics for software safety is

i. The percentage of software hazards out of the system hazards is given by ( no of software hazards / no of system hazards)*100
ii. The average of the software-related hazards is given by (HASCNC + HASCSC + HNSCASC) / (HNSCC + HASCNC + HASCSC + HNSCASC)
iii. The average of the software hazard causes is given by (SC-NSC + SCASC) / (NCNSC+ SCNSC+ SCASC+ NCASC).
iv. The average of Non-software hazard causes with software controls is given by NCASC / (NCNSC+ SCNSC+ SCASC+ NCASC).
v. The average of non-software causes contains software controls is given by NCASC / (NCNSC + NCASC)
vi. The average of the causes contains software controls (SCASC + NCASC) / (NCNSC+ SCNSC+ SCASC+ NCASC).

Software metrics are well-known indicators for software quality analysis [10], [11]. Software metrics are used to measure the software quality of system. Software metrics can be divided according to their usage. Some metrics are software in hazards, causes, and controls metrics and some of them are program metrics used for measuring characteristics of system (i.e. number of lines of code, number of methods etc.). The number of software-related causes and controls of hazards are used to quantify the significance of software. Software safety metrics are used to provide quality assurance on the state of software safety.
Preliminary Hazard Analysis (PHA) Preliminary Hazard Analysis is performed at very stage of software life cycle. It is used to identify the device level concern, safety critical areas, potential hazards, and perform initial risk assessment. The objectives for the system and software hazard analysis are [13], identifying critical system modules and program sections, verifying that software required to handle the failure modes identified by systems/subsystems hazard analysis does so effectively, allowing more rigorous methods and controls to be selected and applied to areas of software which are most critical to the safety of the system, identifying and evaluating safety hazards associated with the software, with the aim of either eliminating them or assisting in the reduction of associated risks, identifying failure modes that can lead to an unsafe state and making recommendation for changes, determining the sequence of inputs which could lead to the software causing an unsafe state and making recommendations for changes.

Software Failure Mode and Effects Analysis (SFMEA) Software Failure Modes and Effects Analysis (SFMEA) is a software-engineering process that investigates the potential effects of postulated software failures on a system and its environment. When the criticality of the effects is also considered, the technique is called a Software Failure Modes, Effects and Criticality Analysis (SFMECA). SFMEA and SFMECA are primarily used to discover software design defects during software development. SFMEA is a structured, table-based process of discovering and documenting the ways in which a software component can fail and the consequences of those failures. The input to a SFMEA is a specification of the design or detailed requirements of the software.

Fault Tree Analysis (FTA) Fault Tree Analysis is an engineering activity that investigates the potential causes of a fault or hazard. FTA is widely used to discover design defects during the development of a system and to investigate the causes of accidents or problems that occur during system operation. FTA may be described as an analytical methodology that uses graphic symbols to visually display the analysis process [11]. Fault tree analysis begins by identifying an undesirable event, known as the top event, associated with a system. The events that could cause the occurrence of the top event are generated and connected by logic operators such as AND and OR.

Event Tree Analysis (ETA) Event Tree Analysis Similar to fault tree analysis, except the analysis starts from a primary event and traces forward to identify its consequences. ETA is a widely used technique used for determining the consequences of a potentially hazardous event.

Failure Checklists (FC) FC produces a list on software failures by applying experience of daily functions and past incident [16], [3]. The main task of FC is to identify the potential hazards in the system. Checklists are a form of safety analysis. As an example, a cochlear implant system is safety critical system and checklist is a simple form of safety analysis. They are useful for understand and examination of system.

DESIGNING FAIL SAFE SYSTEMS Once a hazard has been identified and the associated risk has not been found to be acceptable, the next task is to design the system so that the hazard is eliminated or reduced or controlled. In designing safety, some approaches to reducing risk are effective than other. The system safety design order of precedence for mitigating identified hazards as follows. Hazard elimination through design selection: The best solution to reduce the risk is to eliminate the hazard, either by eliminating the hazardous state or consequences. Design safety could include substituting a hazardous content with a nonhazardous one, simplifying or modifying the design and style, or reducing the amount of hazardous content. Incorporation of safety features or devices (with periodic functional checks of the devices): If hazard cannot be eliminated, then the next best solution is to use safety features and devices. These devices could be active features, such as elevator braking systems or interlocks, or passive systems. Use of warning devices to detect the hazard condition and to produce warning signals: If design selection or safety features and devices are not feasible, then warning devices are the next best solution. Warning devices alert personnel to the presence of a hazardous condition. An example of a warning device is a flashing light at a railroad crossing when the train is approaching. Devel-opment of operating procedures and training for personnel: Procedures and training are used when other methods are not available. Procedures may include the use of personal protective equipment. The next best solution is to control the hazard. Interlocks, lockins, or lockouts are examples of the hazards control. Placement of warning labels, warming lights and buzzers are used for risk mitigation. Labels and warning lights are good design contributes to safety which includes all communications with the user. Redundancy is used to enhance safety in process; here the design becomes more complex. Software which controls critical function in system needs special changes in code and data.

Ensuring Safety during Design Process Even with a safe design, it is possible to increase or decrease device safety, depending on how the software is written. The coding issues for safety are language selection and use of safe coding styles. Languages that provide strong compile-time and run-time checking are considered safer than those that do not.

Exceptions provide a valuable tool for improving safety. Some of the issues to think about are:

- **Language choice**
- **Compile-time checking**
- **Run-time checking**
- **Exceptions vs. error codes**
- **Use of safe language subsets (e.g. avoiding void*)**

Some of the proven rules and guidelines for writing safety-critical code proposed by Gary Holtz [9] are:
Cochlear Implant System as MCPS

A Cochlear Implant System (CIS) is a real-time embedded computing device that can provide a sense of sound to people who are deaf or profoundly hearing-impaired [6], [20], [19]. Significant speech recognition can be achieved with commercially available multichannel cochlear implant systems. Indeed, majority users of cochlear implant system (Bionic Ear) can converse over the telephone for everyday communications. Cochlear Implants are not available to most of the worlds deaf and severely hearing impaired people due to the high cost. Many researchers are attempting to develop low cost but effective cochlear implant devices [5], [2].

The Cochlear Implant system contains four hardware modules which are the following:

i. Digital Speech Processor System (DSPS)
ii. Impedance Telemetry Monitoring System (ITMS)
iii. Implantable Receiver Stimulator (IRS)
iv. Headset Cable (HC)

APPLICATION OF SOFTWARE SAFETY AS CPS

Cochlear Implant System as MCPS

A Cochlear Implant System (CIS) is a real-time embedded computing device that can provide a sense of sound to people who are deaf or profoundly hearing-impaired [6], [20], [19]. Significant speech recognition can be achieved with commercially available multichannel cochlear implant systems. Indeed, majority users of cochlear implant system (Bionic Ear) can converse over the telephone for everyday communications. Cochlear Implants are not available to most of the worlds deaf and severely hearing impaired people due to the high cost. Many researchers are attempting to develop low cost but effective cochlear implant devices [5], [2].

The Cochlear Implant system contains four hardware modules which are the following:

i. Digital Speech Processor System (DSPS)
ii. Impedance Telemetry Monitoring System (ITMS)
iii. Implantable Receiver Stimulator (IRS)
iv. Headset Cable (HC)

trodes and their respective TCL and MCL. ITMS is designed based on programmable logic device FPGA (Field Programmable Gate Array). The ITMS used to send the impedance request to IRS (Implantable Receiver Stimulator) and to receive impedance values with electrode or channel numbers respectively. The function of ITMS is to measure electrode impedances and neural response of a patient through stimulation and recording of electrical signals can facilitate device fitting and parameter adjustments.

Digital Speech Processor System DSPS receives an external sound or speech and generates encoded speech data bits for transmission to IRS via radio frequency transcutaneous link for exciting the electrode array by continuously executing speech/sound processing program embedded in DSPS. The speech processor is a small computer worn on your body and connected to the headset by cable. It receives sounds from a microphone, converts them to electrical signals and sends the signals to the headset. Body-Worn Speech Processor is DSPS. This is one of the three major units (Speech Processor with headset cable, Impedance Telemetry Unit, and Implantable Receiver Simulator unit with electrode array) of the Cochlear Implant System.

Implantable Receiver Stimulator IRS is to stimulate auditory nerve system with the help of electrode array placed inside the cochlea of deafened person. IRS receives directions from the speech processor by way of magnetic induction sent from the transmitter and also IRS receives its power through the transmission. The IRS unit of Cochlear Implant consists of a RF receiver coil made up of Two turns Pt-Ir, Electrode Array with a Reference Electrode, Magnet, and an ASIC shown as in Fig-2. The centerpiece is an ASIC (Application Specific Integrated Circuit) chip, which performs critical function of ensuring safe and reliable electric stimulation. Inside the ASIC chip, there are a forward pathway, a backward pathway, and control units.

Safety Analysis on CPS of CIS

The main function of CPS are

i. Patient information management
ii. Impedance measurement of patient active electrodes contacts with respect to reference electrode
iii. Adjust sound processing and stimulation parameters in a speech processor.
iv. Loading/Mapping THL/MCL of Active Electrodes into Speech Processor.
v. The database application for CPS to manage patients information

The main functions of speech processor are

a. Speech Signal Processing
b. Speech Data Encoding
c. Transmission of encoded speech data to Implanted Receiver Stimulator (IRS) via transcutaneous RF inductive link

Impedance Telemetry Monitoring System ITMS is used finding the active electrodes of electrode array of 12 Electrodes. Implementation of clinical programming software without error plays an important role in the development of different modules such as maintaining patient information, impedance measurement, and determination of stimulation parameters (fitting) and mapping THL/MCL of active electrodes into Speech Processor. A failure is caused in CPS software because of a bug and may alter the external behavior of the program. The initial hazard identification process performed on the CPS identified a number of general system hazards including:

Figure 1: Components of CIS
Wrong patients treatment history retrieved.

Current treatment profile appended to wrong patients record.

Identifying incorrect electrode failure impedance module.

Measurements of impedance values are incorrect.

Wrong calculation of active electrode values.

Faulty decision in CPS regarding ITMS malfunctioning.

Identification of active electrodes are wrong regarding ITMS malfunctioning.

Release of incorrect volume.

Incorrect calculation of THL, MCL values, volume delivered to wrong location.

Incorrect calculation of THL, MCL values, volume too high.

Finding THL, MCL for failed electrodes.

Communication failure between CPS and DSPS or ITMS.

Volume sends to wrong delivery location.

Control program software is corrupted.

The SFMEA, a sample of which is shown in the Table 1 below presents some software failure modes defined for CPS. We identify the possible critical events for other top level hazards derived from the SFMEA analysis. Figure 2 shows the sample fault tree for the top event incorrect release of volume. The faults identified using FTA are

Ft to fail THL, MCL value calculation.

Fault in DSP.

Logic Fault in CPS Software.

Implementation Fault in CPS Software.

The database application faults for CPS to manage patients information

The Causes and Controls for each Hazard

The categories of causes and controls are mentioned for each hazard and this report is used for finding the metrics for CPS. The causes line is the total number of causes in the hazard report. The data were calculated for all hazards in CPS project. Approximately found a total of 52 hazards from CPS project, 638 causes, and 1670 controls.

Metrics for MCPS

Metrics are used to measure the importance of software (CPS) with respect to system safety. Software causes and controls are analyzed according to their specificity, whereas verifications of software controls are evaluated according to the level of confidence or assurance in the verification.

Recommendations for Design and Coding for CPS

From the safety analysis we have conducted, the major critical events that might occur and the corresponding safety properties the CPS software has to implement, and which are controlled by the software in the CIS are listed below.

<table>
<thead>
<tr>
<th>Table I: software failure modes defined for cps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Mode or hazard</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>Release of volume too high</td>
</tr>
<tr>
<td>Volume sends to wrong delivery location</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II: CPS hazard table</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.No.</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table III: CPS cause table</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.No.</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>Transferred Causes</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table IV: cps control table</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.No.</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>Total Controls</td>
</tr>
</tbody>
</table>
The software shall receive accurate patient data input values. If CPS receives wrong patients treatment history, it will calculate current treatment profile incorrectly which could lead to hazards associated with impedance measurement, fitting and mapping functions.

The software shall verify all ITMS connectivity, electrode values before the start of the calculation of impedance values.

The software shall calculate THL, MCL values of active electrodes. Failure to do so can lead to injury to healthy tissue.

The software can load THL, MCL values of active electrodes in DSPS. If CPS loads incorrect THL, MCL values in DSPS, Injury to healthy tissue.

### CPS FUNCTIONAL OPERATION

The CPS software is designed for DSPS and ITMS used by an audiologist for performing post operative fitting procedure for better recognition of sound. The program contains multiple functional modules such as patient information management, UART Settings, impedance measurement, fitting and mapping. The software is designed under VB.net2008, with a database MSACCESS. The designed database tables are responsible to record patient basic information, medical record, and evaluation of hearing abilities, evaluation of speech and language status, rehabilitation status, evaluation of psychological status, medical and audio logical evaluation, processor programming, and specific training with processor accessories and so on. The following figure shows the partial block diagram of CPS

Initially, the CPS starts from Audiologist registration and then goes for patient registration. Whenever the CPS displays "invalid details please input all the required details, Stop the CPS process and proceed to again registration process. Whenever the ITMS is loaded it reads the impedance values from impedance database table if available and displays each channel resistance value in corresponding textboxes and displays the resistance values in chart control also. If the resistance values are not yet stored it displays the error message as insufficient recipient data please try again. The impedance measurement modules display the impedance values if the audiologist had already measured the impedance values of the patient in normal text form and also graphical representation. If the impedance values are not available it displays the null values. Whenever fitting module is loaded it reads the impedance values from the impedance table and displays in the corresponding textboxes of each channel and creates a new row in mapping table with default values of TCL and MCL. The audiologist may vary TCL as well as MCL of particular channel by moving the dynamic range bar upwards or downward and then press Stimulation button.

### Table V: measure the importance of software with respect to system safety

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Metrics</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The software hazards in the system hazards</td>
<td>42%</td>
</tr>
<tr>
<td>2</td>
<td>The software hazards related to software</td>
<td>51%</td>
</tr>
<tr>
<td>3</td>
<td>The average of the software hazard causes in the system hazards</td>
<td>10%</td>
</tr>
<tr>
<td>4</td>
<td>The average of Non-software hazard causes with software controls</td>
<td>12%</td>
</tr>
<tr>
<td>5</td>
<td>The average of non-software causes contains software controls</td>
<td>18%</td>
</tr>
<tr>
<td>6</td>
<td>The average of the causes contains software controls</td>
<td>22%</td>
</tr>
<tr>
<td>7</td>
<td>The average of causes is transformed</td>
<td>38%</td>
</tr>
<tr>
<td>8</td>
<td>The average of controls is transferred</td>
<td>22%</td>
</tr>
<tr>
<td>9</td>
<td>The average of the non-transformed hazard controls in specific software controls</td>
<td>10%</td>
</tr>
<tr>
<td>10</td>
<td>The average of non-transferred controls are references to generic software controls</td>
<td>5%</td>
</tr>
<tr>
<td>11</td>
<td>The average of all inner methods of the class, which use particular instance variables</td>
<td>26%</td>
</tr>
<tr>
<td>12</td>
<td>The centis the number of other classes coupled to the analyzed one</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>The number of immediate subclasses of a given class</td>
<td>23</td>
</tr>
<tr>
<td>14</td>
<td>The centis the number of lines of code except for the lines containing comment only</td>
<td>5000</td>
</tr>
</tbody>
</table>
Software Safety Properties and Observed Behavior of CPS

Through the experimentation analysis, the following things have been observed:

i  For the new patient, follow all the steps to identify the implant and electrode type. Do not make any assumption. Violation of this requirement will cause misidentification problem.

ii  Do not edit or alter files generated or used by the Fitting Software. Because it loaded into database when edit that change is not loaded into database. That means backup of patient data in the database is not automatically provided. It is the responsibility of the user/audiologist to backup patient data files.

iii Only CPS software should run in our PC not other programs. Violation of this requirement will cause software conflict. Solution is shut down all programs except CPS and sees if the situation improves.

iv CPS can be effective tool, but they must be maintained and used correctly otherwise it gives security problems. v The CPS software does not require calibration, service or maintenance. But periodic updates will be provided by the developer for compatibility to the latest technology updates.

vi The display of Hardware not found message appears after patient selection if the user tries to open a task (Fitting, Telemetry or DSPS) and no appropriate hardware is found by CPI because of following reasons those are speech processor switched off or ITMS unit switched off or all cables are not connected properly or The batteries of the speech processor and/or the ITMS unit are low or A cable is defective or The PCs hardware is defective.

vii The display of Connection to hardware failed message appears after patient selection if the user tries to open a task (Fitting, Telemetry or DSPS) and no appropriate hardware is found by CPI because of following reasons those are speech processor switched off or ITMS unit switched off or all cables are not connected properly or The batteries of the speech processor and/or the ITMS unit are low or A cable is defective or The PCs hardware is defective.

viii Data Mismatch message appears if the selected patient does not correspond with the patient data stored in the speech processor.

ix Upload failed or was incomplete message appears if The serial connection was interrupted or The Log file shows further details about the failed upload or Repeating the upload is possible by closing the Fitting windows and opening again.

x Download failed or was incomplete message appears if the serial connection was interrupted or The Log file shows further details about the failed download or Download can be repeated by pressing the relevant download button again.

xi The display of an error message indicates a severe problem with the software and/or the operating system. In the case of an error message, it will probably be necessary to quit the CPI software and resolve the problem. It is recommended to remove the coil from the implant site if this occurs.

CONCLUSION

The identification of these metrics used to improve the software safety of medical devices such as CPS of CIS. In this paper we identified software hazards, causes and controls for safety analysis and found that more than 50% caused by software or software was involved in the identification of hazardous condition. By analyzing hazard reports, software safety analysis process identified some risks in the CPS of CIS. This process supports to improve the software safety process by some risks in CPS.

REFERENCES


