DATA-DRIVEN RESILIENCY ASSESSMENT OF MEDICAL CYBER-PHYSICAL SYSTEMS

BY

HOMA ALEMZADEH

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Ravishankar K. Iyer, Chair
Professor William H. Sanders
Professor Lui R. Sha
Professor Jaishankar Raman, Rush University

ABSTRACT

Advances in computing, networking, and sensing technologies have resulted in the ubiquitous deployment of medical cyber-physical systems in various clinical and personalized settings. The increasing complexity and connectivity of such systems, the tight coupling between their cyber and physical components, and the inevitable involvement of human operators in supervision and control have introduced major challenges in ensuring system reliability, safety, and security.

This dissertation takes a data-driven approach to resiliency assessment of medical cyber-physical systems. Driven by large-scale studies of real safety incidents involving medical devices, we develop techniques and tools for (i) deeper understanding of incident causes and measurement of their impacts, (ii) validation of system safety mechanisms in the presence of realistic hazard scenarios, and (iii) preemptive real-time detection of safety hazards to mitigate adverse impacts on patients.

We present a framework for automated analysis of structured and unstructured data from public FDA databases on medical device recalls and adverse events. This framework allows characterization of the safety issues originated from computer failures in terms of fault classes, failure modes, and recovery actions. We develop an approach for constructing ontology models that enable automated extraction of safety-related features from unstructured text. The proposed ontology model is defined based on device-specific human-in-the-loop control structures in order to facilitate the systems-theoretic causality analysis of adverse events. Our large-scale analysis of FDA data shows that medical devices are often recalled because of failure to identify all potential safety hazards, use of safety mechanisms that have not been rigorously validated, and limited capability in real-time detection and automated mitigation of hazards.

To address those problems, we develop a safety hazard injection framework for experimental validation of safety mechanisms in the presence of accidental failures and malicious attacks. To reduce the test space for safety validation, this framework uses systems-theoretic accident causality models in order to identify the critical locations within the system to target software fault injection.

For mitigation of safety hazards at run time, we present a model-based analysis framework that estimates the consequences of control commands sent from the software to the physical system

through real-time computation of the system's dynamics, and preemptively detects if a command is unsafe before its adverse consequences manifest in the physical system.

The proposed techniques are evaluated on a real-world cyber-physical system for robot-assisted minimally invasive surgery and are shown to be more effective than existing methods in identifying system vulnerabilities and deficiencies in safety mechanisms as well as in preemptive detection of safety hazards caused by malicious attacks.

To my parents for their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisers, Prof. Ravishankar K. Iyer and Prof. Zbigniew Kalbarczyk, for their tremendous support and invaluable advice during various stages of the development of this dissertation. Ravi always encouraged me to pursue exciting new directions, pushed me to step beyond the established boundaries, and gave me the courage to put forward controversial ideas. Zbigniew generously provided me with thorough feedback and advice on different aspects of my work. They have been great friends, mentors, and role models and have taught me the essential skills of becoming an independent researcher and an effective teacher.

I would also like to thank the members of my doctoral committee, Prof. Jaishankar Raman, Prof. Lui Sha, Prof. William H. Sanders, and Prof. Janak H. Patel for their excellent advice, encouragement, and feedback throughout my Ph.D. work. Prof. Raman has been a great mentor and collaborator, who helped my work have real societal impact by introducing me to several important problems in medicine and in particular to the interesting field of minimally invasive robotic surgery.

The research described in this dissertation could not have happened without the help of many wonderful colleagues and collaborators in the DEPEND group, the Coordinated Science Laboratory, and the Healthcare Engineering Systems Center at the University of Illinois. I especially thank Daniel Chen, Xiao Li, Prof. Thenkurussi Kesavadas, Prof. Ditlev Monrad, Prof. Zhanpeng Jin, Qingkun Li, Hui Lin, Mushfiq U. Saleheen, Raymond Hoagland, Valentin Sidea, Dr. Nithin Nakka, Dr. Lelio Di Martino, Key-whan Chung, Zack Estrada, Phuong Cao, and Skylar Lee for their help in developing and implementing the ideas in this dissertation and for many insightful research discussions. I am deeply grateful to Heidi Leerkamp and Laurie Fisher for their kind assistance with many administrative tasks, and to Jenny Applequist, Carol Bosley, Fran Baker, and Jan Progen for their patience in reviewing my papers and this dissertation.

I also thank Prof. Blake Hannaford, David Drajeske, Andrew Lewis, and the other members of the Biorobotics Lab and the Applied Dexterity Inc. at the University of Washington for providing access to the RAVEN II surgical robot in their lab and for their continued support, kind feedback, and many useful discussions. I also thank Prof. Nancy G. Leveson and the members of the MIT

Partnership for a Systems Approach to Safety (PSAS) for their invaluable advice and great feedback on my work on systems-theoretic safety analysis of surgical robots.

I am also thankful to many dear friends who accompanied me on this unforgettable journey, in particular Mehrnoush, Gabriela, Joana, Dina, and Safa, for their warm friendship that created enjoyable moments and gave me breaks from long hours of work.

I dedicate this dissertation to my parents, Nasrin and Hadi, who are my role models of honesty, strength, and persistence. I owe a debt of gratitude to them and to my dear brother, Hassan, for their everlasting love and unswerving support over the years.

Last but not least, thank you Farzad, for your unconditional love, endless patience and understanding, continued encouragement, and faithful support of my academic endeavors.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1. Motivation

Advances in computing, networking, and sensing technologies have resulted in the ubiquitous deployment of cyber-physical systems in safety-critical settings such as aerospace, energy, transportation, and healthcare. In particular, there is significant deployment of medical cyber-physical systems in various clinical and personalized settings, ranging from tiny implantable pacemakers and defibrillators to wearable health monitoring devices, complex patient monitors in intensive care units (ICUs) and expensive systems for radiation therapy and robot-assisted surgery.

The increasing complexity and connectivity of medical systems, the tight coupling between their cyber and physical components, and the inevitable involvement of human operators in supervision and control have introduced significant challenges in ensuring system reliability, safety, and security. During 2007–2013, over 6.8K medical device recalls and 2.4 million adverse events (including 923K injuries and 49K deaths) were reported to the U.S. Food and Drug Administration (FDA) [1]. Nearly 24% of the recalls were due to computer-related failures in software, hardware, batteries, or system interfaces, which affected more than 18 million medical devices on the market [2], [3], incurring medium-to-high risk of severe health consequences to patients and considerable costs for manufacturers and caregivers.

Even though state-of-the-art medical systems are often designed with safety mechanisms that attempt to detect failures and put system into safe states, in practice, they have limited capability in real-time detection and automated recovery from safety hazards. Many medical devices are recalled because of design flows that missed comprehensive hazard analysis and risk assessment, or use of safety mechanisms that were not rigorously validated for resiliency. The recovery mechanisms are often limited to recommendations on manual actions to be taken by the device operators or costly repair or replacement of the devices by the manufacturers [2].

1.2. Challenges

This section presents some of the challenges in design and assessment of resilient medical cyber-physical systems. These challenges motivate the research presented in this dissertation.

1.2.1. Analyzing Incident Causes and Impacts

Medical device incidents are reported by healthcare professionals, device manufacturers, and investigators to the U.S. Food and Drug Administration (FDA) [1]. A major component of these reports are human-written narratives describing adverse events, reasons for recalling a device, or corrective actions taken by the manufacturers. Analysis of those reports provides valuable insights on the causes and impacts of real incidents experienced in the field and how the designs of devices could be improved in the future. However, automated extraction of safety-related semantics from the reports is a challenging task, requiring interpretation of free-form natural language text, which is often abstract and inconsistent. Further, without models to describe the complex interactions within the system and between the system and operators, and without considering domain-specific semantics and underlying context, it is difficult to automatically infer all the incident causes from the reports.

Previous work on analysis of the FDA data mostly relied on manual review of incident reports and keyword searching to identify specific types of problems (e.g., software faults or security exploits) [4] – [9]. These approaches require a significant amount of human effort and may still produce inaccurate results and non-comprehensive demonstration of real problems.

1.2.2. Safety Assessment

Medical cyber-physical systems are embedded with interconnected electronic components and software modules that interact with mechanical/physical components, multiple human operators, and patients, under stringent timing constraints. With a variety of internal and external disturbances and changes in dynamics of patients and the environment, comprehensive safety assessment of the system as a whole is a challenging task.

Standard hazard analysis and safety assessment techniques recommended and used by the medical device industry (e.g., Fault-Tree Analysis (FTA) [10] and Failure Mode and Effect Analysis (FMEA) [11]) primarily focus on the failures of individual components or human errors

in the system. Other important factors that contribute to system safety, such as complex software errors, unsafe component interactions, and contextual factors (the system states and conditions under which actions are taken) are often not thoroughly considered during typical hazard analyses [12]. Therefore, many incidents occur because the safety hazards and security vulnerabilities were not identified and removed during the design process, or because the resiliency of detection and mitigation mechanisms are not adequately validated. Further, with the increasing size and complexity of software, even if the most rigorous risk analyses and validation processes are used, some residual failures manifest as unforeseeable hazards during the device operation [2].

1.2.3. Mitigation of Safety Hazards

Although medical systems are often embedded with safety mechanisms that detect failures and put the system into a safe state, in practice, timely detection and effective mitigation of safety hazards (caused either by incidental failures or malicious attacks) are very challenging. This is because of the complex nature of incidents, the difficulty of accurately modeling the causality relationships [12], and uncertainties in operator actions and patient health status.

Mitigation of safety hazards requires preemptive detection of the adverse consequences of commands and understanding of the semantics of interactions among operators and cyber and physical components at different layers of the system.

Existing detection and recovery mechanisms are designed primarily based on inclusion of redundancy and consistency checks or monitoring the software and physical system components separately from each other. The interactions among operators, software modules, and physical components, the dynamics of the physical system, and the patient status are often not considered in the design of those mechanisms, mostly because of cost and real-time constraints.

1.3. Contributions

This dissertation addresses the aforementioned challenges by taking a data-driven approach to resiliency assessment and design for resiliency of medical cyber-physical systems. The overall contribution of this dissertation is a methodology that combines data analytics, systems-theoretic accident causality analysis, safety hazard injection, and dynamic-model based state estimation to

assess system resiliency and derive design of safety monitors that can make quantitative measurements of the system's safety and mitigate hazards in a timely manner.

We assert that in medical cyber-physical systems, resiliency can be best achieved through:

- Large-scale analysis of past incidents to enhance our understanding of the multidimensional causes involved in incidents and to obtain statistically confident measures of their impacts.
- Experimental validation of safety and security protection mechanisms in the presence of realistic safety hazard and security attack scenarios.
- Design of mechanisms for continuous monitoring of human operators, cyber and physical system states, and patient status, and rapid detection of events that can lead to safety violations.

In particular, we focus on resiliency assessment and design of protection mechanisms for robot-assisted surgical systems used in minimally invasive surgery [13], as an example of safety-critical cyber-physical systems with multiple humans in the loop. We present a simulation platform that integrates the following:

- *A systems engineering approach to safety* (adapted from [12]), which uses a causality model based on systems and control theories to analyze safety-critical events and identify hazardous system states and their potential causes based on the hierarchical control structure of the surgical robot.
- *A robotic surgical simulator*, which leverages the robot control software and enables modeling of the behavior of human operators and the dynamics of robotic hardware and mechanical components (and potentially tool-tissue dynamics).
- A *safety hazard injection engine* that performs detailed injections into robot control software (i) in a surgical simulator to emulate the impact of safety hazards, without causing adverse impacts on the electrical and mechanical components of the real robot, or (ii) in the actual robot to conduct resiliency assessment and validation of the safety mechanisms in a real-world system.
- A *dynamic-model based analysis framework* that estimates the consequences of interactions between the cyber and physical components by real-time measurement and computation of the robot's dynamics, and preemptively detects safety hazards to issue efficient and timely mitigation/recovery actions.

This platform is built based on the results of our study on causes of adverse events and reasons for recalls of robot-assisted surgical systems. The proposed solution can potentially be applied to a broader class of safety-critical medical and cyber-physical systems that involve humans in the loop, including electric power grid and transportation systems.

The main contributions of this dissertation can be summarized as follows:

- **Analysis of computer failures in medical devices:** We present MedSafe, a framework for large-scale analysis of structured and unstructured data from the public FDA databases on medical device recalls and adverse events. By combining techniques from natural language processing and statistical learning, MedSafe enables discovery of safety issues that originated from computer failures in medical devices with an average accuracy of 91%. We characterize the computer failures in terms of fault classes, failure modes, and recovery actions taken by the manufacturers, and measure their impact on the patients and the manufacturers in terms of severity of hazards and the number of device repairs and removals. Our analysis of over 13K recall records using MedSafe shows that although software is the major cause of failures in computer-based medical devices, hardware, battery, and I/O failures have much larger impact in terms of the number of recalled devices and the cost of device removal/repairs. We also identify several classes of safety-critical medical devices that posed serious hazards to patients and provide many examples of devices that were designed without appropriate handling of safety issues or rigorous validation of their safety mechanisms.

- **Systems-theoretic analysis of adverse events:** We develop an approach for constructing ontology models based on the human-in-the-loop control structures used in system-theoretic accident causality analysis [12] to formalize the semantic interpretation of incident narratives. The proposed ontology model enables automated extraction of important safety-related features from the unstructured text to provide deeper understanding of multidimensional causes of incidents. We demonstrate the effectiveness of system-theoretic analysis of adverse events in a case study of more than 10K adverse events reported for robotic surgical systems used in minimally invasive surgery. We characterize those events by identifying the system hazards and potential causal factors that led to unsafe control actions during robotic procedures as well as inadequate safety mechanisms in both system design and operational practices that led to health-threatening events (injuries and death).

We also analyze the trends of adverse events and patient impacts across the years and surgical specialties.

- **Safety hazard simulation for resiliency assessment and safety training:** Driven by the insights gained from analysis of real incidents, we propose a technique for experimental resiliency assessment of system in the presence of potential safety hazards. This framework uses software fault injection to emulate the safety hazards caused by accidental failures or malicious threats targeted at different layers of the system control structure. In order to reduce the test space for safety validation, the systems-theoretic accident causality models [12] are used to identify the critical locations within the cyber-physical system to target with software fault injection. We demonstrate the effectiveness of this technique by validating the safety mechanisms in the RAVEN II robot, an open-source platform for research in telerobotic surgery [14], [15]. Our results show that with a reduced number of targeted fault injections, we can identify several vulnerabilities in the safety mechanisms, which, if not removed, might lead to adverse consequences for the patient and physical system. We also demonstrate the application of the safety hazard injection framework to the simulation of real hazard scenarios (extracted from the FDA data) in a virtual environment for simulation-based safety training of robotic surgeons.

- **Cyber-physical attacks on control systems of surgical robots:** We introduce a new family of targeted attacks on the control systems of teleoperated surgical robots that strategically inject malicious control commands into the system at a critical time during surgery. We illustrate these attacks by implementing prototype malware targeting the RAVEN II robot and show that they can lead to unforeseen and abrupt jumps of a few millimeters in the robot manipulators within only a few milliseconds or the unavailability of the system due to an unwanted transition to a halt state. We discuss the detectability of the attacks and that their adverse consequences can be best mitigated by continuous monitoring of both cyber and physical components and by estimating the physical consequences of control commands. We also discuss other possible ways that the availability of commercial surgical robots can be compromised on a wider scale, by discussing vulnerabilities in the open-source robotic middleware and the servers used for remote diagnostic services during surgery.

- **Preemptive detection of safety hazards caused by incidental failures or malicious attacks:** We present a dynamic-model based analysis framework that estimates the consequences of control commands sent from the cyber domain to physical domain through real-time computation of the system's dynamics to preemptively determine if a command is unsafe before the actual execution of the command progresses and its adverse consequences manifest in the physical system. The proposed framework can be used for timely and effective mitigation of safety hazards due to either accidental failures or malicious attacks. We evaluate the detection accuracy of the framework using two real attack scenarios implemented on the RAVEN II robot. Our experiments show that the dynamic-model based analysis framework achieves better performance than the existing safety mechanisms that rely solely on monitoring of the control commands in the cyber domain, without considering their semantics in the physical system.

1.4. Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 presents the design of MedSafe and our large-scale analysis of computer-related failures in medical devices. Chapter 3 describes our proposed ontology model for systems-theoretic analysis of adverse events and the case study of adverse events in robotic surgery. Chapter 4 describes our safety hazard simulation framework for resiliency assessment and safety training, which is demonstrated on the RAVEN II robot. Chapter 5 presents the targeted cyber-physical attacks on the control system of teleoperated surgical robots and the dynamic-model based analysis framework for real-time detection and mitigation of safety hazards. Finally, Chapter 6 concludes the dissertation and presents directions for future research. Appendix A discusses concerns regarding underreporting in collection and analysis of data. Appendix B provides a full description of example adverse event reports from the FDA MAUDE database, which are discussed in Table 3.3. Appendix C describes our previous work on design of resilient patient monitoring devices, serving as preliminary results for one of the directions for future work suggested in Chapter 6.

CHAPTER 2
COMPUTER FAILURES IN MEDICAL DEVICES

2.1. Overview

The U.S. Food and Drug Administration (FDA) regulates medical devices sold in the United States by requiring manufacturers to follow a set of pre- and post-market regulatory controls. Medical devices are classified by the FDA into 5,853 distinct types and 19 medical specialties, such as anesthesiology, cardiovascular, clinical chemistry, general hospital, general surgery, and radiology, indicating their regulatory class and marketing requirements. All devices must comply with general controls such as registration, listing, quality system, and labeling, but device classes require additional special controls. Devices such as pacemaker chargers, patient beds, or chairs which do not pose a very serious risk to health of patients are exempt from special regulatory controls. Most of the devices with medium level of risk such as patient monitors, robotic devices, and diagnostic software require a special set of controls, including submission of a Premarket Notification 510(k) to FDA. The 510(k) Premarket Notification must demonstrate that the device is substantially equivalent to one legally in commercial distribution in the United States. Devices such as pacemakers, radiation therapy equipment, and infusion pumps that pose a significant risk of illness or injury to patients are required to go through a Premarket Approval (PMA) process in which claims made for their functionality should be proved by submission of clinical data [1].

After a medical device is distributed in the market, FDA collects and monitors the reports of adverse events and other problems with the devices, and alerts health professionals and the public, by issuing recalls or safety notifications, to ensure proper use of devices and patients' health and safety.

The FDA data on medical device recalls and adverse events of medical devices provide valuable insights on the past failures of medical devices and how the designs could be improved in the future. However, the automatic analysis of these reports is a challenging task, requiring semantic

---

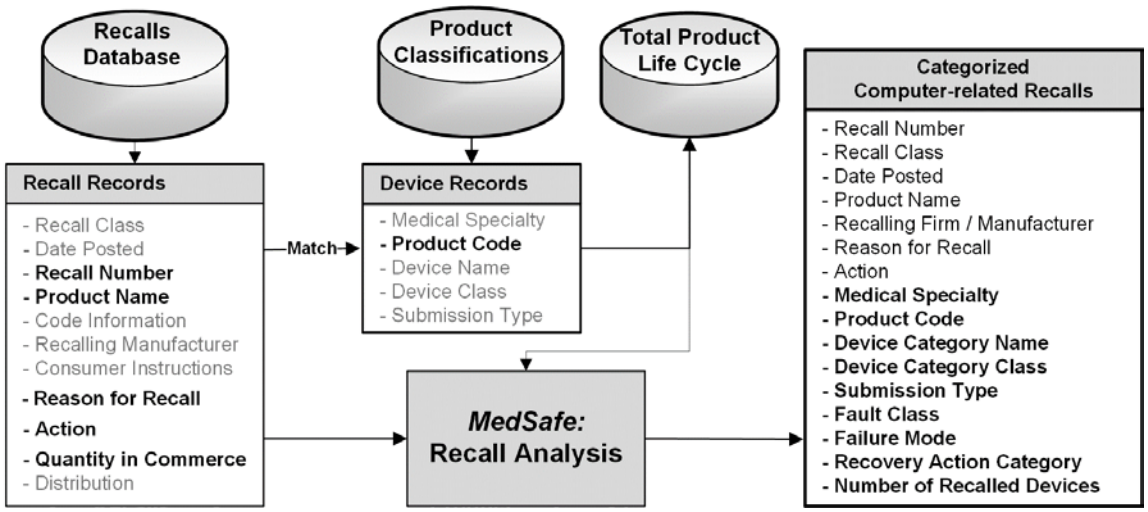
This chapter contains material from the previously published works [2], [3], coauthored with Z. Kalbarczyk, R. K. Iyer, J. Raman, and R. Hoagland, copyrighted by IEEE.

interpretation of natural language text and consideration of all potential causes and contextual factors, including faulty components, human errors, and system states.

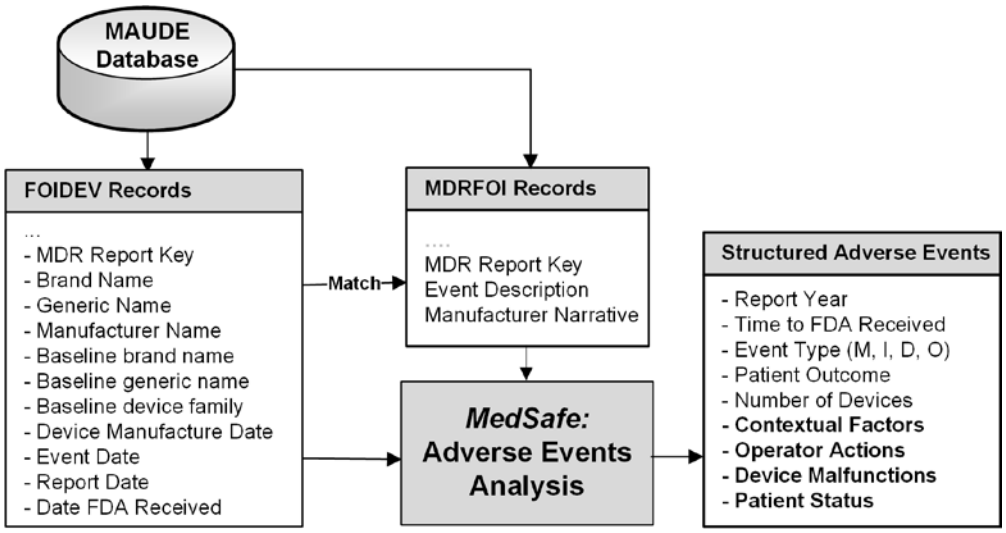
We developed MedSafe, a framework for automated analysis of data on failures and safety incidents of medical devices. It enables the analysis of larger sets of reports from the FDA databases, in order to provide deeper understanding of the causes of failures and safety incidents in various types of medical devices and statistically confident measures on how the failures and safety issues impact the patients.

MedSafe combines state-of-the-art techniques in natural language processing, machine learning, knowledge discovery, and accident analysis, to extract structured information from the unstructured human-written descriptions of recalls and adverse events data. Figure 2.1 shows the input and output for recalls and adverse events analyses in MedSafe, which comprises the following steps:

1. **Data extraction, filtering, and coalescing**:
   - Collect the recalls and adverse events data from the online FDA databases for any desired period of time, device type, or manufacturing company.
   - Filter non-relevant records and coalesce duplicate records related to the same events.
   - Extract additional information on the recalled or faulty devices, by cross-referencing recall records and adverse event reports with the product classification and device-related data.
2. **Recalls data analysis:**
   - Identify recalls related to computer-based medical devices.
   - Characterize computer-related recalls in terms of fault classes, failure modes, recovery actions taken by the companies.
   - Measure the impact of failures on the patients and the manufacturers in terms of severity of potential hazards and the number of device repairs/removals.
3. **Ontology modeling for accident analysis:**
   - Create a device-specific ontology model based on the control structure of the device by identifying the main control loops, the controllers, the controlled processes, the interactions among them, the contextual factors, and control flaws.
   - Define a set of tags and semantic dictionaries for each class in the ontology model and a set of pattern matching rules for identifying and annotating the tags in the natural language text.



(a)



(b)

Figure 2.1. MedSafe framework: (a) Recalls data analysis, (b) Adverse event data analysis.

4. Adverse event data analysis:

- Semantically parse the adverse event descriptions by ontology-driven annotation of the natural language text with safety-related features (e.g., operator actions and device malfunctions) to facilitate system-theoretic causal accident analysis of adverse events.

Section 2.2 provides an overview on the data sources used in our analysis. Section 2.3 presents our preliminary analysis of recalls and adverse events to identify safety-critical computer failures in medical devices. In Section 2.4 the MedSafe recalls data analysis is described. The MedSafe adverse event data analysis is presented in more detail in Chapter 3. The source code for MedSafe is publicly available for use by the other researchers [16], [17].

2.2. Data Sources

In what follows we provide a brief overview on the following three databases used by the FDA for collection of pre- and post-market data on medical devices: the Medical & Radiation Emitting Device Recalls (Recalls) database; the Manufacturer and User Facility Device Experience (MAUDE) database, and the Total Product Life Cycle (TPLC) database [1].

2.2.1. Medical Device Recalls

The FDA's Recalls database contains classified medical device recalls since November 1, 2002. A recall is a voluntary action that a manufacturer, distributor, or other responsible party takes to correct or remove from the market any medical device that violates the laws administrated by the FDA. Recalls are initiated to protect the public health and well-being from devices that are defective or that present health risks such as disease, injury, or death. In rare cases, if the company fails to voluntarily recall a device that presents a health risk, the FDA might issue a recall order to the manufacturer.

The FDA classifies recalls into three classes based on the relative degree of health hazard the device presents. *Class I* recalls indicate that there is a reasonable chance that use of the device will cause serious adverse health problems or death. Examples of recalled devices in this class are defibrillator/monitors, anesthesia machines, and drug delivery systems. *Class II* indicates devices that might cause temporary or medically reversible adverse health consequences or pose a remote chance of serious health problems. Glucose monitors and multi-parameter patient monitors are examples of devices recalled in this class. *Class III* indicates devices that violate the laws administrated by the FDA but are not likely to cause adverse health consequences, such as patient programmers and ultrasound system software.

Each record submitted to the FDA's recalls database contains structured information such as a unique *Recall Number*; the *Recall Class*; *Date Posted* on FDA, as well as unstructured narratives describing the *Product Name* (i.e. the name of device and a description of its functionality), *Recalling Firm/Manufacturer*, *Quantity in the Commerce* (i.e. number of devices distributed on the market), and important information on the safety issues of the device, *Reason for Recall* and

Class 2 Recall - Vivid E9 ultrasound system	
Date Classified	November 21, 2013
Recall Number	Z-0373-2014
Product	GE Vivid E9 ultrasound system. GE Vivid E9 is a Track 3 diagnostic ultrasound system, which is primarily intended for cardiac imaging and analysis, but which also includes vascular and general radiology applications. The Vivid E9 incorporates a variety of electronic array transducers operating in linear, curved linear, sector/phased array or matrix array format, including two dedicated CW transducers and several real time 3D transducers. It consists of a mobile console with keyboard control panel; color LCD/TFT touch panel, LCD color video display and optional image storage and printing devices. It provides high performance ultrasound imaging and analysis and has comprehensive networking and DICOM capability.
Recalling Firm/ Manufacturer	GE Healthcare, LLC
Manufacturer Reason for Recall	GE became aware of a potential safety issue due to a system slow down and system lock up associated with the software of Vivid E9 ultrasound system. During a critical procedure the Vivid E9 Scanner may gradually become unresponsive and eventually lock up, with the result that the vivid E9 Scanner must be rebooted.
Action	An Urgent Medical Device Correction letter, dated 10/11/2013, was sent to 3 departments within the affected hospitals. The letter described the safety issue, and identified affected devices. The letter stated to reboot the scanner in an unresponsive /lock up condition. Also if the system is being used for an interventional procedure, the TEE probe should remain in the patient while system is rebooting. A GE healthcare service engineer will install a correction to affected devices.
Quantity in Commerce	710
Distribution	Worldwide and U.S.

Figure 2.2. A sample record from the FDA Recalls database, available at [18].

the recovery *Action* taken by the manufacturer to address the device problems. Figure 2.2 shows a sample record from the recalls database [18]. There are several challenges in analyzing the recalls data:

(i) The recalls data is accessible online through the FDA website, but all the fields of the records are not provided in the downloadable files. Also, the *Quantity in the Commerce* field contains a combination of textual and numeric information which should be parsed for calculation of the total number of devices distributed in the market that are affected by a recall.

(ii) Many of the recall records represent the same failure event reported for different devices or device models manufactured by one company. These recalls have the same *Manufacturer* fields and very similar *Reason for Recall* and *Action* fields, while their *Quantity in the Commerce* fields contain different values.

(iii) Only a fraction (22.9%) of the recalls are related to failures of computer-based medical devices, i.e. the computer failures of medical devices. The identification of these records requires the semantic interpretation of natural language text in the *Product Name*, *Reason for Recall*, and *Action* fields of the records.

(iv) Recalls data does not include the FDA device classification information such as the *Medical Specialty*, *Product Code*, *Device Type*, *Device Class*, and *Submission Type*, which could provide important insights on the regulatory process used for the approval of recalled devices and understanding the impact of device failures in different medical specialties.

2.2.2. Medical Device Adverse Event Reports

The MAUDE database is a publicly available collection of *suspected medical device-related* adverse event reports, submitted by mandatory (user facilities, manufacturers, and distributors) and voluntary (healthcare professionals, patients, and customers) reporters to the FDA. FDA regulations require firms that receive complaints to notify the FDA of medical device incidents, including device malfunctions, serious injuries, and deaths associated with devices. Not all reported adverse events lead to recalls; manufacturers and the FDA regularly monitor adverse events to detect and correct problems in a timely manner.

Each adverse event report contains information such as: unique *MDR Report Key*; device *Brand Name* and *Generic Name*; *Event Type* indicating the impact of adverse event on the patient ("Malfunction," "Injury," "Death," or "Other"); *Device Problem* indicating the problem from a list of problem categories provided by the FDA; and human-written *Event Description* and *Manufacturer Narrative* fields that provide a short description of what actually happened, as well as any comments made or follow-up actions taken by the manufacturer to detect and address device problems or indications on whether the adverse event was actually related to a device malfunction.

2.2.3. Total Product Life Cycle Data

The Total Product Life Cycle (TPLC) database integrates pre-market data on medical devices, including device classifications, pre-market approvals (PMA), and pre-market notifications (510(k)), with post-market data, including adverse events and recalls. Each record provides the pre-market review information and a list of adverse events and recalls reported for a device type.

2.3. Safety-Critical Computer Failures in Medical Devices

We first performed a preliminary study on the causes of *computer-related failures* in medical devices and their impact on patients, by analyzing human-written descriptions of recalls and adverse event reports, obtained from the FDA databases. We define *computer-related failure* as any event causing a computer-based medical device to function improperly or present harm to patients/users, due to failures in any of the following components of the device: *software*, *hardware*, *I/O*, or *battery*. Other failures of computer-based devices that could not be easily categorized in these four classes are classified in *other* category.

In-depth study of *recall data* allowed us to characterize the computer-related failures based on:

- ***fault class*:** the defective components that led to device failure
- ***failure mode*:** the impact of failures on the device's safe functioning
- ***recovery action category*:** the type of actions the manufacturer took to address the recall
- ***number of recalled devices*:** the quantity of recalled devices distributed on the market
- ***device category*:** the categories and types of recalled devices

We used the overall number of devices that are affected by each recall, as a metric to measure the impact of failures. We specifically focused on *safety-critical recalls* that include:

(i) Recalls classified by FDA as Class I, presenting a high likelihood of severe injury or death to patients

(ii) Recalls for which the *Reason for Recall* field specifically indicated a patient *safety* issue such as *injury* or *death*

(iii) Recalls for which the *Reason for Recall* field explicitly indicated potential of exposing patients/users to immediate *"Physical Safety Hazards"* such as *"overdose"*, *"overexposure"*, *"electrical shock"*, *"burning"*, and *"fire"* (classified under *"Physical Safety Hazard"* failure mode in Section 2.3.2.2).

Safety-critical recalls were used as a basis to find categories and types of *safety-critical medical devices,* whose failures will most likely lead to life-critical consequences for patients. Analysis of *adverse event reports* allows us to measure the impact of device failures in terms of actual adverse consequences (e.g., serious injuries or deaths) reported to the FDA. Finally, based on specific safety issues identified for life-critical medical devices, we discuss the challenges in design of next-generation medical devices.

2.3.1. Analysis Flow

Figure 2.3 shows the overall analysis flow, which started with extraction of 13,413 recall records from the database, reported to the FDA from January 1, 2006 to December 31, 2011.

In **Step 2**, we identified the computer-related recalls by analysis of the *Reason for Recall* and *Action* fields of records in the FDA recalls database. Those fields contain human-written unstructured text explaining the main reason for the recall and recovery actions taken by the manufacturer to address the recall. Many of the recall records have the same reasons because the same component or part is used in different devices or models manufactured by the same company. After coalescing recall records that indicated the same manufacturing company and the same reason for recall in their descriptions, we came up with 5,294 unique *Recall Events* (40% of the total recall records) or what we refer to as *Recalls* in the FDA database.

In **Step 3**, we used text normalization and part-of-speech tagging (using Natural Language Toolkit (NLTK) [19]) to automatically extract the most frequently used adjectives and nouns in the "Reason for Recall" description of the recalls. This list was then manually reviewed to create a dictionary of 461 common computer-related keywords that could potentially represent failures of computer-based devices. The list of computer-related keywords was further categorized into the

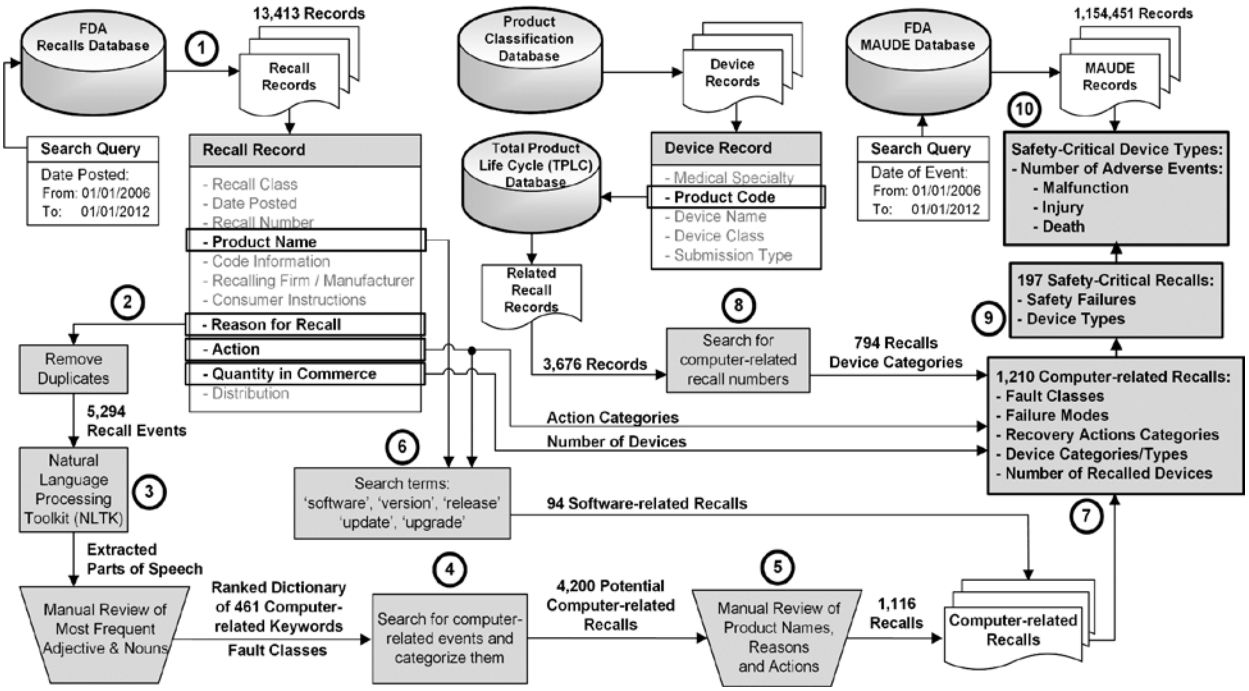


Figure 2.3. Methodology for analyzing safety-critical computer-related recalls.

classes of Software, Hardware, Battery, Input/Output (I/O), and Other, corresponding to defects in different components of the device.

In **Step 4**, the extracted dictionary was then used to identify computer-related recalls by searching for keywords in *Reason for Recall* descriptions. That led us to a reduced list of 4,200 potential computer-related recalls, whose corresponding recall records were manually reviewed for validation and further categorization.

In **Step 5**, we manually reviewed and excluded many of the records from the list of computer-related recalls because their *Product Name*, *Reason for Recall* and *Action* fields did not indicate a computer-based device recall. The final list of computer-related recalls included 1,116 unique recall events.

In **Step 6**, we also found 94 additional computer-related recalls reported because of software errors (software-related recalls) that were missed in our reason analysis process because the human-written explanations of reasons did not include any terms from the dictionary related to computers. We extracted these additional recalls by searching for the terms *'software'*, *'version'*, and *'release'* in the *Product Name* field and terms *'software'*, *'update'*, and *'upgrade'* in the *Action* fields.

In **Step 7**, by manual review of the computer-related recalls, we extracted *Fault Class*, *Failure Mode*, *Recovery Action Category*, and *Number of Recalled Devices* for each recall. The number of recalled devices was calculated by summing up the quantities listed in the recall records related to each recall event. For example, in Table 2.1 the fourth recall event was reported in five records in the recalls database, which together affected a total number of 7,152 devices on the market. In some instances where the total number was entered in all the recall records related to a recall event, we only counted it once.

We then used the FDA TPLC database which integrates the information such as device name, type, category (medical specialty), and regulatory class of recalled devices with a subset (3,676) of recall records. We extracted that information for 794 of computer-related recalls in our study and then used it as a training set for finding the names, types, and categories of the rest of computer-related recalls.

Finally, we obtained a total of 1,210 computer-related recalls that affected an overall number of 12,024,836 devices distributed in the United States and worldwide. The 1,210 recalls were used as the basis for deriving statistics on *Fault Classes*, *Failure Modes*, and *Recovery Actions* of

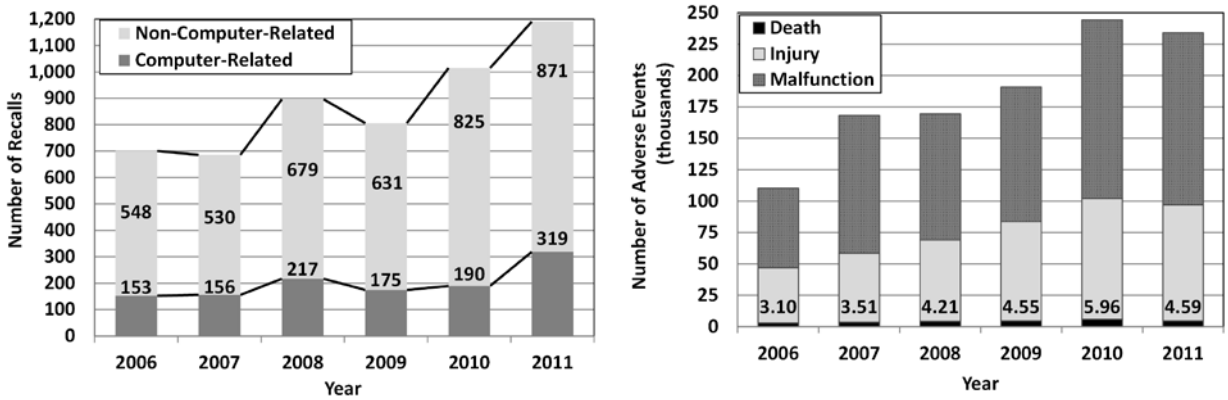
Table 2.1. Examples of computer-related recalls.

Fault Classy	Example Computer-related Recall Events					Num. of Records	Num. of Devices
	Year	Class	Reason for Recall	Summary of Action	Failure Mode		
Software	2008	2	The product has a software interface problem. When connected to the main system it will not allow the system to recognize the instrument which makes the instrument nonfunctional at all sites. Risks associated are loss of operability of the instrument; delay in surgery; and loss of dexterity.	-Urgent Device Recall letter was issued to customers, instructing them to return the product. -Customers were instructed to segregate the product in a secure area for customer service representatives.	Device Operation Failure	1	11
Hardware	2007	1	Failure to Deliver Shock; a defective capacitor may cause the delay or non-delivery of the defibrillating shock which may result in failure to resuscitate the patient	-The firm issued alerts and instructions to customers on how to return their device. -The firm will exchange the recalled defibrillator with a replacement and new five (5) year warranty.	Treatment Interrupt/ Therapy Failure	1	1,794
Other	2010	1	Potential for the device to power off then on by itself; or to power off by itself and requiring the operator to turn it back on; or the device doesn't turn off.	-The firm issued an "Urgent Medical Device Correction" notification and advised customers to keep the affected device in service and to test the units in accordance with operating instructions. -A service visit was scheduled within 60 days.	Device Operation Failure	1	3,609
Battery	2008	1	Unintentional rebooting: Pump products exhibit an intermittent loss of power due to intermittent loss of contact between battery cap and battery canister resulting in the device resetting. The failure of battery cap may result in failure of the device to administer insulin therapy which may result in hyperglycemia.	-The recalling firm issued notification letters to the patients with insulin pumps to inform them of the problem and that they needed to replace the battery.	Treatment Interrupt/ Therapy Failure	5	7,152
Input/ Output	2010	2	Speakers on the patient monitors may fail; causing absence of an audible alarm and delaying patient treatment.	-The firm sent notification letters and instructions to customers on actions to take while awaiting their replacement speaker assemblies. -Affected products may continue to be used provided that the user routinely checks for the display of the "Speaker Malfunction" INOP at power-up. If this INOP is experienced or there is no sound from speaker, remove device from use and contact the service representative.	Alarm/ Message Error	2	21,654

computer-related failures (Section 2.3.2), to identify safety-critical medical devices, their specific safety issues, and patient impact (Section 2.3.3), and to provide insights on the challenges in design of next-generation medical devices (Section 2.3.4).

2.3.2. Data Analysis Results

During the study period, over 5,290 recalls and 1,154,450 adverse events were reported to the U.S. Food and Drug Administration (FDA). As Figure 2.4 shows, since 2006, there was a 69.8 percent increase in the number of recalls and a 103.3 percent increase in the number of adverse events, reaching approximately 1,190 recalls (see Figure 2.4(a)), 92,600 patient injuries, and 4,590 deaths in 2011 (see Figure 2.4(b)). The number of computer-related recalls almost doubled, reaching an overall number of 1,210 (22.9 percent of all recalls) in 2011. Previous studies conducted during the periods of 1983–1997 [4] and 1999–2005 [5] attributed, respectively, 383 (six percent) and 1,261 recalls (33.4 percent) to software-based medical devices.



(a) (b)

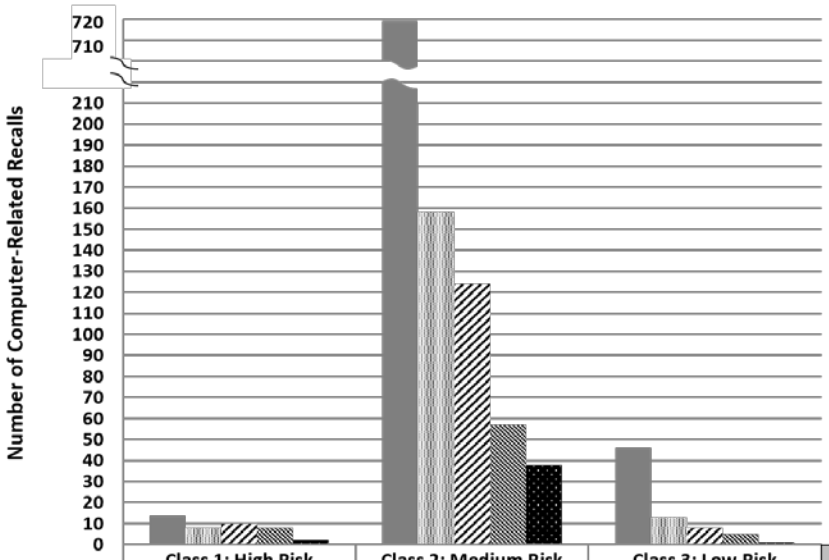
Figure 2.4. (a) Total number of recalls per year (2006-2011): Computer-related and non-computer-related, (b) Total number of adverse events (2006-2011): Malfunctions, Deaths, and Injuries (Numbers on the bars indicate number of deaths in thousands per year).

2.3.2.1. Fault Classes

Table 2.2 lists example keywords from the dictionary used to identify computer-related failures in each fault class. The "Software" class represents failures due to software errors. The "Hardware" category includes both electrical issues and defects of internal circuits, while the "Input/Output (I/O)" category includes failures due to sensors, connections, display, or speakers. The "Battery" category represents defects in batteries, power cords, or power supply units that might cause interruption/failure of computer-based device function or cause harm to patients. "Battery" failures were included as computer-related failures because a typical safety-critical computer system should be able to detect, respond, and manage such failures and prevent harm to patients. The "Other" category includes recalls whose descriptions indicate a computer-related failure, but are not sufficient to be categorized in any of the above categories.

Table 2.2. Example dictionary keywords.

Fault Class	Example Dictionary Keywords
Software	software, application, function, code, version, backup, database, program, bug, java, run, upgrade
Hardware	board, chip, hardware, processor, memory, disk, pcb, disk, electronic, electrical, circuit, leak, short-circuit, capacitor, transistor, resistor
Other	error, system, fail, verification, self-test, reboot, web, robotic, calculation, document, performance, workstation
Battery	battery, power, supply, outlet, plug, power-up, discharge, charger
Input/ Output	sensor, alarm, message, screen, signal, interface, monitor, connect, button, scanner, key, speaker, wireless, terminal, communication



	Class 1: High Risk	Class 2: Medium Risk	Class 3: Low Risk	Total Recalls	Num. of Devices
Software	14 (33.3%)	718 (65.6%)	46 (75.3%)	778 (64.3%)	2,303,441 (19.2%)
Hardware	8 (19.0%)	158 (14.4%)	13 (27.4%)	179 (14.8%)	4,228,133 (35.2%)
Other	10 (23.8%)	124 (11.3 %)	8 (12.3%)	142 (11.7%)	2,831,048 (23.5%)
Battery	8 (19.0%)	57 (5.2%)	5 (6.8%)	70 (5.8%)	2,385,613 (19.8%)
Input/Output	2 (4.8%)	38 (3.5%)	1 (2.7%)	41 (3.4%)	276,601 (2.3%)
Total Recalls	**42 (3.5%)**	**1095 (90.5%)**	**73 (6.0%)**	**1210**	**12,024,836**

Figure 2.5. Distribution of computer-related recalls in fault classes and risk levels. The last column of the table shows the total number of devices on the market affected by the recalls in each fault class.

Table 2.1 shows example recall records categorized in each fault class. Figure 2.5 illustrates the distribution of recalls across different fault classes and recall classes (risk classes). The following are our observations based on these results:

- Note that all Class I recalls are classified as safety-critical according to criterion (i) in the beginning of Section 2.3. Our analysis shows among Class I recalls, 42 were due to computer-related failures (Figure 2.5, column 2). Software failures accounted for 33.3% (14) of Class I recalls, while Hardware (8), Other (10), Battery (8), and I/O (2) combined were the reason for 66.7%. Clearly, a non-negligible fraction of computer-related recalls are due to non-software-related failures.
- The majority (90.5%) of computer-related recalls were classified by FDA as Class II, with a medium risk of health consequences. Of these, we classified 66 to be safety-critical based on criterion (ii) for safety-critical recalls introduced in the beginning of Section 2.3. In each case the manufacturer's description explicitly indicated that the device failure resulted in or had the potential to result in a patient safety issue, injury, or death.

- When we simply look at the overall number of recalls, similar to what other studies (e.g., [4] – [7]) reported, software is a major cause (14.7%) of the total recalls. Additionally, 64.3% of computer-related recalls are due to software failures. However, we get a very different perspective by considering the total number of devices on the market that were impacted by specific recall types (software, hardware, other, battery, and I/O). This analysis can be derived from the last column of Figure 2.5. If we look at the total number of devices, hardware-related recalls had a larger impact (almost 84% more) compared to software. Of all the recalled devices on the market, 57.3% were recalled because of hardware, battery, or I/O failures, and only 19.2% because of software faults.

2.3.2.2. Failure Modes

To show the breadth of failures that might impact the safe functioning of a computer-based medical device, we group the failures under six different categories shown in Table 2.3. Number of recalls in different FDA recall classes categorized in each failure mode along with example failures of each category is shown in the table. For example, 84 of 1,210 computer-related recalls were due to failures affecting the alarm functionality of the device and were grouped under the "Alarm/Message Error" category.

We draw the attention to safety-critical recalls identified based on criterion (iii) in the beginning of Section 2.3. For these 91 recalls, devices had potential to expose patients/user to immediate safety hazards (e.g., overdose, electrical shock, and fire), and are grouped under the "Physical Safety Hazards" failure mode. It is interesting that nearly all physical safety hazards were in Class II, but it is important to consider them as safety-critical because the manufacturer's description explicitly indicated a possibility of immediate harm to patients/users.

The last three columns of Table 2.3 show example recalls in each failure mode category that are classified as safety-critical according to the criteria defined in Section 2.3 and will be further discussed in Section 2.3.3. In each example, information used for identification of the safety-critical recall is highlighted in bold in the reason description and the corresponding criteria are shown in the last column. For example, the third recall is related to a hardware defect that might lead to loss of the system pump and injection of hot fluid into patient's uterus. Although this recall was classified in Class II, it was selected as a safety-critical recall by criterion (ii).

Table 2.3. Computer-related failure modes.

Failure Mode	Recalls Count/ Class			Example Failures	Example Safety-Critical Recalls		
	I	II	III		Recall Class	Recall Record Number: Reason for Recall Summary	Criteria
Alarm/ Message Error	4	76	4	- Alarm reset - Lack of audible alarms - Missed alarms - Unexpected/false alarms	I	Z-0051-2012: Pumps stop infusing, backup alarm sounds; but the "Run" LEDs advance as if the pumps were infusing.	(i)
Physical Safety Hazard	2	89	0	- Electrical shock - Smoke/fire/explode - Unintended movement - Overdose/over exposure	II	Z-0119-2009: A short-circuit (e.g., in a cable or the control units) can result in **uncontrolled and unstoppable movement** of the Video Fluoroscopy table. This failure might lead to serious deterioration in state of patient health.	(iii)
Display/ Image Error	1	156	11	- Blank image - Display freeze - Image distortion/corruption - Loss of image data	I	Z-0006-2011: Under certain wireless network conditions a communication error can occur that freezes the PC Unit screen. This failure may result in delay of therapy and serious **injury or death.**	(i) (ii)
Treatment Interruption/ Therapy Failure	18	129	3	- Delayed/failed shock delivery - Infusion/ventilation failure - Signal analysis failure - Loss of monitoring	II	Z-0689-2007: Defective integrated circuit board could result in loss of the system pump and **patient injury (hot fluid 90 degree C into uterus)**.	(ii)
Device Operation Failure	12	234	23	- Device inoperable - Failure at startup - Failure to stop exposure - Hangup/Freeze	II	Z-1474-2009: Unusual occurrence of system lockups of cardiovascular X-ray imaging systems causes image acquisition failure and user has to reset the system. **One patient death has been reported related to this issue**.	(ii)
Calculation/ Output Error	4	311	20	- Corrupted patient files - Inconsistent output - Incorrect calculation/display - Miscalculation	I	Z-0263-2012: Drug dosage calculation may indicate incorrect values; misalignment of ECG-ART waveforms was observed on the central station.	(i)

For 113 of the 1,210 recalls there was not enough details on failure symptoms, or the event could not be classified in any of the defined failure modes.

2.3.2.3. Recovery Actions

We classified the recovery actions taken by manufacturers to five categories shown in Table 2.4. Also Table 2.1 (column 5) shows actions taken for the example computer-related recalls. In the following, the denominators refer to the number of recalls/devices in the specified fault classes:

- For 18.4% (223/1,210) of recalls the recovery action was limited to sending notifications to customers about the device problem, or providing instructions on how to avoid or workaround the problem.
- The majority of computer-related recalls due to software faults (80% = 623/778) were addressed by releasing a new software version or software patch to fix the problem. Sending notifications or instructions was the next most common action to address software-related recalls (16.7% = 130/778).
- For hardware-related recalls, in most cases, customers were required to completely remove the device and/or return it to the company for replacement (36.3% = 65/179), or the device or part of it had to be corrected/repaired by the company (38.5% = 69/179). Interestingly, 4.5% (8/179) of hardware-related recalls were addressed by a software update.
- Of all the devices affected by the recalls, approximately 17.8% (2,145,087/12,024,836) required replacement of parts or a complete removal. Additionally, the majority of these replacements were because of battery (52.9% = 1,135,478/2,145,087) or hardware (37.6% = 805,868/2,145,087) failures.

For 10% (121/1,210) of the records, the *Action* field information was not available or sufficient to categorize them.

Table 2.4. Recovery action categories and examples.

Recovery Action Category	Example Recovery Actions	Recalls Count
Safety Notification	"Consignees were notified by letter on/about December 1, 2005."	223
Safety Instructions	"In the notice letter, Agfa HealthCare is also providing customers with the recommended workaround. The workaround is to only print from the Viewer screen or to print the ECG once confirmed. The Viewer screen, however, does not allow the user to print batches of reports as does the Index screen."	
Software Update	"The letters stated that the recall was to the user level and requested that the user perform the software upgrade, which will eliminate the possibility of shock and burn."	632
Repair	"The notice asks that the customers inspect their units for signs of discoloration indicative of a faulty connector. The customers were instructed to return the product to CSZ for repair by contacting their Customer Service division and obtaining a Return Authorization number and specific instructions concerning packaging and returning of the unit(s) for repair."	95
Replace/ Remove	"The letter indicates that the firm will exchange the recalled defibrillator with a replacement and new five (5) year warranty."	139

These results show the importance of non-software-related (e.g., hardware and battery) failures in terms of higher cost for manufacturers, caregivers, and patients. For example, for implantable cardioverter-defibrillators recalled during 1990–2000, a total cost of $870 million including device checks/analyses ($83 million) and replacements ($787 million) was estimated [8]. These costs could be considerably reduced by the use of fault-tolerance techniques to enable recovery from such failures without requiring complete removal of the devices.

2.3.3. Safety-Critical Medical Devices

In the final stage of analysis we focused on safety-critical devices whose failures present the highest likelihood of severe life-critical consequences to patients (Step 9 in Figure 2.3). A total of 197 (16.3%) computer-related recalls were identified as safety-critical, including (i) 42 Class I recalls, (ii) 66 Class II recalls whose *Reason for Recall* field specifically indicated a patient *"safety"* issue such as *"injury"* or *"death"*, and (iii) 89 Class II recalls with a *"Physical Safety Hazards"* failure mode. Those 197 recalls together affected 2,447,894 devices on the market.

We found that the majority (80.7% = 159/197) of safety-critical recalls were for devices used in Radiology (e.g., linear accelerators), Cardiovascular (e.g., automated external defibrillators), General Hospital (e.g., infusion pumps), Anesthesiology (e.g., ventilators), and General Surgery (e.g., electrosurgical accessories). More importantly, 73.8% (31/42) of Class I recalls were for Cardiovascular and General Hospital devices such as defibrillators, patient monitors, and infusion pumps. Almost all those devices were approved by the FDA under a medium level of regulatory controls (510(k) clearance).

Table 2.5 shows example types of safety-critical medical devices that were recalled because of potential harm to patients. The total number of computer-related recalls, safety-critical computer-related recalls, and example fault classes and failures for each device type were extracted from the recalls database. The last three columns of Table 2.5 present the number of adverse events reported for these devices in the MAUDE database. We obtained these numbers by searching for the devices based on their *Names* and *Product Codes* in the MAUDE database (Step 10 in Figure 2.3). For extraction of computer-related adverse events, we used the *Product Problems* of the reports.

Table 2.5. Safety-critical medical devices: Computer-related recalls and adverse event reports.

Device Category	Device Type (Product Codes)	Safety Critical Computer-Related Recalls				Number of Computer-Related Adverse Events		
		Num of Recalls	Num of Devices	Example Faults Classes	Example Failures	Death	Injury	Mal-function
Radiology	Linear Accelerator (IYE)	13	4,415	Software	- No interlock of beam delivery - Unexpected gantry rotation - Incorrect treatment plan - Dose delivered to wrong location	0	0	5
				Other	- Unexpected flat panel movement - Unexpected couch movement			
	Image Processing System (LLZ)	15	15,069	Software	- Mismatch/wrong image orientation - Inaccurate annotation/data print - Unintended images displayed - Incorrect/incomplete data display - Overestimated image scales	1	0	4
	Image-Intensified Fluoroscopic X-Ray System (JAA)	7	3,468	Software	- Unexpected system lockup - Inaccurate detection - Incorrect dose exposure - Unstoppable X-ray exposure - Image storage failure	0	1	2,186
Cardiovascular	Defibrillator: External (Non-Wearable) (MKJ)	17	415,537	Hardware Battery	- Delayed/failed shock delivery - Energy discharge failure	16	1	281
				Software	- Premature shutdown - Incorrect energy/shock delivery			
				Other	- Unexpected power on/off			
	Defibrillator: Implantable (NIK/LWS/ MRM)	2	170,542	Software	- Loss of rate response - Premature battery depletion - Loss of telemetry - Aborted therapy	293	14,281	11,028
	Implantable Pacemaker/ Pulse Generator (DXY/LWP/NVZ)	1	40,164	Hardware	- Loss of rate response - Premature battery depletion - Loss of telemetry	60	3,301	2,742
	Physiological Patient Monitor/ Arrhythmia Detector/Alarm (MHX/DSI)	10	38,394	Software	- Incorrect dosage - Misaligned waveforms displayed - Delayed audible alarms - Failure to restart - Burn or electrical shock hazard	4	79	276
General Hospital	Infusion Pump (FRN/LZH/ LKK/MEA)	15	945,300	Software	- Incorrect safety alarms	23	574	2,399
				Hardware Battery	- Delayed/over/under infusion - Infusion failure without alarms - Electrical shock, burn, fire hazard			
	Insulin Infusion Pump (LZG)	2	13,756	Battery	- Insulin delivery failure - Unexpected shutdown w/o warning	0	4	15

Out of 75,267 identified computer-related adverse events, around 50% (representing 398 deaths, 18,241 injuries, and 18,937 malfunctions) were related to the devices shown in Table 2.5. However, our observation is similar to other studies [6], [9] that there are inaccuracies and underreporting in the MAUDE database and inconsistencies between MAUDE and Recalls databases. As an example of the inconsistency, we see that although safety-critical computer-related recalls affected a significant number of devices used in Radiology, very few severe adverse events due to computer problems could be identified for these devices in the MAUDE database. Nonetheless, implantable pacemakers, defibrillators, and infusion pumps dominate the computer-related failures (35 recalls) and fatalities (392 deaths). This observation can be explained by the large number of these devices in use, for treatment of critical conditions such as sudden cardiac arrest.

2.3.4. Discussion

By relying on complex software, sophisticated hardware, batteries, sensors, and network communications, future medical devices face several challenges in terms of reliability, safety, and security: Increased complexity raises the possibility of component interaction accidents (e.g., the first recall in Table 2.1); portability makes the devices more vulnerable to power outages (e.g., the fourth recall in Table 2.1); and interconnectedness increases the chance of error propagation (e.g., the third recall in Table 2.3) and facing failure storms that devices will not be able to handle in a fail-safe manner.

Additionally, medical devices are prone to major security and privacy vulnerabilities such as unauthorized control of sensing and communication functions of devices and access to private patient data. Despite the significance of challenges related to security and privacy, these issues are severely underreported in the FDA databases. According to [9], 142 instances of malware infections affecting medical devices were detected between 2009–2011, but none of them were reported in the MAUDE database. Our analysis of FDA data found three adverse events related to computer malware and virus infections in a defibrillator, a radiology workstation, and an imaging system reported by the manufacturers and user facilities, and one voluntary report of unauthorized access to a glucose monitor. We only found one FDA recall related to computer malware affecting an imaging system and categorized it under the fault class of “Other”.

Our study found that:

- While software failures remain the major cause (64.3%) for recalls of computer-based medical devices, hardware, battery and I/O are also significant contributors to failures that can lead to potential life-critical hazards.
- Hardware, battery, and I/O failures had a larger impact (affected 6.5 million, 54%, of recalled devices on the market) in terms of the number of devices affected by the recalls (almost three times) and the cost of device removals/repairs.
- Of all the safety-critical recalls, 80.7% were for devices used in radiology (for instance, linear accelerators), cardiovascular (e.g., automated external defibrillators), general hospital (e.g., infusion pumps), anesthesiology (e.g., ventilators), and general surgery (e.g., electrosurgical accessories). More important, 73.8% of Class I recalls were for cardiovascular and general hospital devices, such as defibrillators, patient monitors, and infusion pumps, for which the highest number of device-associated fatalities are also reported.
- In many cases the chain of events leading to failures could not be identified based on the limited information provided in the recalls descriptions. However, by looking at example safety-critical failures studied here (e.g., hardware defect that might lead to injection of hot fluid into patient's body shown in Table 2.3) we see that many of the recalled devices were either designed without identifying and handling the safety issues or the safety mechanisms were not designed/implemented correctly.

These results emphasize the importance of designs with well-defined safety requirements and implementations that employ robust error detection techniques and fail-safe mechanisms that are rigorously validated. In what follows we discuss major challenges in design of next-generation safety-critical medical devices.

2.3.4.1. Hazard and Risk Analysis

International standard for risk management of medical devices (ISO 14971 [20]) and the FDA require the manufacturers to maintain a process for identifying foreseeable hazardous situations of the device (see [21] for example hazard categories identified for infusion pumps), estimating the risks associated with each hazard, and controlling the risks by defining the safety requirements

(see [22] for example safety requirements for a generic infusion pump model) and implementing effective risk control measures.

However, traditional safety analysis techniques recommended by the FDA [23] and commonly used by the industry (including Fault-Tree Analysis (FTA) [10], Event Tree Analysis [24], Hazard and Operability Study (HAZOP) [25]) and reliability analysis techniques (e.g., Failure Mode and Effect Analysis (FMEA) [11]), mostly focus on the reliability of individual components in the system or performing probabilistic risk assessment. Those techniques have limited capability in identifying other contributing factors to the system safety such as complex software errors, component interaction failures, human errors, complex human decision-making, and flawed management in the design [12]. For example, the first recall in Table 2.1 was due to an interaction failure between the device and an instrument that made the instrument nonfunctional and presented the risk of complications during surgery. This can be considered as a hazardous situation that was not foreseen in the hazard analysis phase.

In [12] a new hazard analysis technique, called System Theoretic Process Analysis (STPA), for safety-driven design of complex software-intensive systems is proposed. This technique treats the design process as a control optimization problem rather than a component failure problem. The hazardous situations are prevented/mitigated by creating safety constraints based on identifying potentially unsafe control actions in the system and modifying the design to eliminate the effect of the control flaws.

2.3.4.2. Error Detection and Validation

It is clear from the example recalls studied here that some residual faults/errors can escape even the most rigorous risk analysis, design, and validation processes and manifest as failures and unforeseeable hazards during the device's operation. For example, consider a defect in the integrated circuit board that might lead to injection of hot fluid into patient's body (shown in Table 2.3). Such catastrophic failures are probably caused by hard to test corner cases and could be prevented by fault-injection based validation and formal modeling of key failure modes of system. In [26] symbolic fault-injection is shown to be successful in detecting corner cases that might evade error detection in safety-critical air traffic control software.

Alternatively, the failures can be detected at runtime before leading to hazardous situations using techniques such as runtime assertions, watchdog timers, self-test mechanisms and periodic

system checks (e.g., for critical parts such as batteries, sensors, processor, and memory). The study reported in [27] demonstrates use of static program analysis for design of application-specific runtime assertions to detect data errors leading to failures with high coverage at very low cost.

2.3.4.3. Fail-Safe Mechanisms

Full device removal may not be an acceptable option to address device failures because of the high cost for manufacturers, user facilities, and patients. There are many well understood fail-safe mechanisms and failure-recovery techniques used in modern computing systems that can be brought into medical devices in order to manage the failures at lower cost. For example, battery or hardware failures leading to power loss and unexpected shutdowns (such as examples in Table 2.5) could be managed by turning off power to unused components of the system and maintaining power for the critical parts to avoid sudden power outages (e.g., sleep modes are used in modern embedded systems such as cell phones).

Also techniques such as fault containment (used in aerospace and commercial systems) can be used for isolating the faulty units or components (e.g., damaged batteries) and moving the system into a fail-safe mode without presenting harm to patients/users. Intelligent reconfiguration strategies for switching to backup batteries or redundant hardware units in case of failure can be employed. For example, the uncontrolled movement of device (in second recall of Table 2.3) was not stoppable even by manual disconnection of power, because the device automatically switched to a backup battery. In this case, identifying the type of failure and the reason for power loss (whether intentional or accidental) before deciding to switch to a backup battery could stop the unintentional movement and potential patient injury.

2.3.4.4. Recalls and Adverse Events Reporting

FDA mechanisms for reporting recalls and adverse events by manufacturers can assist in preventing future adverse events through lessons learned from the earlier problems. However, current FDA databases for reporting recalls and adverse events suffer from underreporting, inaccuracies, and inconsistencies that make it difficult in many cases to identify the causes of failures and their impact on patients to determine how the design of future devices could be improved. The following are recommendations based on our study on how the reporting mechanisms could be improved:

- Providing robust and systematic interfaces for reporting recalls and adverse events so that:
  - More accurate and complete information (e.g., *Device Name*, *Product Code*, and *Product Problem*) is entered in the reports.
  - List of keywords representing different *Product Problems* in the MAUDE database [1] more precisely reflects causes of device failures, especially computer-related failures.
- Creating integrated databases of recalls and adverse events such that:
  - *Product Name* and *Reason for Recall* fields of each recall record correspond to standard device names, product codes, and device categories defined by the FDA.
  - Recall records can be cross-referenced with related adverse event reports in MAUDE database.

2.3.4.5. FDA's Role in Device Regulation and Approval

FDA guidelines and safety recommendations (e.g., 2010 FDA initiative for external defibrillator improvement [28], 2010 industry guidance for infusion pumps [21], and 2013 guidance for pulse oximeters [29]) to harden life-critical devices recommend the introduction of formal mechanisms for improving the pre-market review and approval of devices. One FDA study [30] introduced the idea of developing usage models for different classes of devices to provide generic safety features and test cases that can be used by manufactures. A more recent idea has involved the use of assurance cases for formal communication of claims, arguments, and evidences about devices from companies to the FDA. In the FDA guidance document for infusion pumps [21], manufacturers are specifically recommended to submit assurance case reports for the approval of devices.

2.3.5. Summary

This study helped us to characterize the causes for computer failures and their impact on patients and to identify the most common types of safety-critical computer-based medical devices which were involved in life-critical incidents. Although a significant part of our analysis involved manual reviewing of the recalls and adverse events data, the extracted computer-related recalls can be used as a training set to automate the analysis of new data submitted to the FDA databases. In Section 2.4, we present our approach for automating the analysis of medical device recalls.

2.4. Automated Classification of Computer-Related Recalls

To address the challenges in analysis of the recalls data (discussed in the Section 2.2.1), MedSafe employs techniques from natural language processing and information retrieval [31] (including the vector-space models for feature extraction, similarity measures from information theory, and machine learning), to automate the (i) collection of recall records, (ii) extraction of unique recall events, and (iii) identification of computer-related recalls, as described next.

We evaluated MedSafe on the recalls data (over 16K records) submitted to the FDA between years 2006–2013. The analysis steps and evaluation results for each step are described next.

2.4.1. Recalls Data Collection

MedSafe first downloads all the recall records submitted for a desired period of time from the FDA online database. The missing information in the records (e.g., the *Main Prodcut Name*, *Quantity in Commerce*, and *Action* fields) are obtained by querying the online database using unique *Recall Number* of each recall record and parsing the corresponding HTML file of the record in the FDA website. Then the number of devices (*Device Quantity*) affected by each recall record is extracted by parsing the *Quantity in Commerce* field, using regular expressions and heuristic rules.

Evaluation: The results of this step were evaluated by manually reviewing recalls submitted to the FDA between years 2007–2013. MedSafe achieved 97.3% accuracy in calculating the total number of devices affected by the recalls. We found that during the study period, a total of 16,881 recall records were reported to the FDA, from which 6,864 (40.7%) were unique recall events.

2.4.2. Recall Events Extraction

As mentioned before, many of the records in the FDA recalls database represent the same failure event reported for different devices manufactured by the same manufacturer. So we extracted the unique recall events related to the same failures through identifying and coalescing duplicate recall records. The duplicate records are identified by measuring the similarity between natural language descriptions provided in the *Manufacturer*, *Reason for Recall*, and *Action* fields.

Figure 2.6 depicts the steps taken by MedSafe for identifying the similar (duplicate) recall records. We first modeled the *Reason for Recall* text of each recall record as a vector r_i of

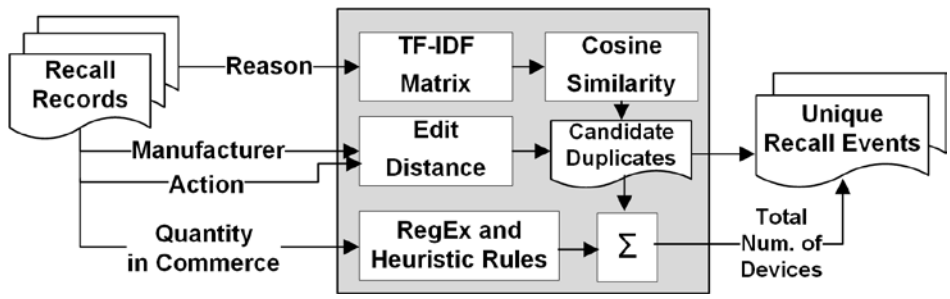


Figure 2.6. Analysis steps for identifying unique recall events.

weights w_k, representing the importance of each keyword in the *Reason for Recall* text of that recall. Then the set d_i of recall records with the most similar *Reason for Recall* fields to the recall record r_i, was obtained by calculating the cosine similarity between all the *Reason for Recall* vectors r_j and r_i and filtering those records that had a similarity of more than a pre-defined threshold (e.g., $th_1 = 0.8$, denoting a similarity of more than 80%):

$$d_i = \{j \big| \langle r_i, r_j \rangle \geq th_1\} \tag{2.1}$$

The final list D_i of candidate duplicate records was then determined by using a Levenshtein (edit) distance measure e to compare the *Manufacturer* (denoted by m_i) and *Action* (denoted by vector a_i) fields of the similar records and filtering those that were initiated by the same manufacturer and were addressed using very similar recovery actions, as follows:

$$D_i = \{j \in d_i \big| e(a_i, a_j) + e(m_i + m_j) \geq th_2\} \tag{2.2}$$

We used an edit distance of less than 20% (th_2 = 0.2, denoting a similarity of more than 80%) to identify similar *Manufacturer* and *Action* fields of candidate and the current records. Finally, all the similar recall records were coalesced into one recall event (denoted by R_i) and the total number of devices affected by the event was calculated by summing up the numeric *Device Quantity* values extracted for each record.

Evaluation: The results of this step were evaluated using subsets of recalls submitted to the FDA between Janaury 2006 and November 2013 (100 records from each year), which were coalesced by manual reviewing of the reports. MedSafe achieved an accuracy of close to 100% in coalescing similar records and calculating the total number of devices. However, the MedSafe speed at this step can be severely affected by scaling to larger number of recalls.

After the publication of this work, the FDA online database has started to provide a unique *Event ID* for each recall record (only for those recalls submitted after 2007), which can also be used as an alternative faster way for identification of the duplicate records.

2.4.3. Recall Classification

At this step, MedSafe uses natural languge parsing in conjunction with statistical learning to automatically identify the computer-related recalls. In Section 2.3, we defined a *computer-related* recall as an event causing a computer-based medical device to function improperly or present harm to patients or users due to failures in device's software, hardware, I/O, or battery. Figure 2.7 depicts the steps taken by MedSafe for identifying the computer-related recalls.

First, the *Reason for Recall* (or *Action*) field of each recall event is normalized by removing the punctuations and English stop words and converting the text into lowercase. MedSafe then

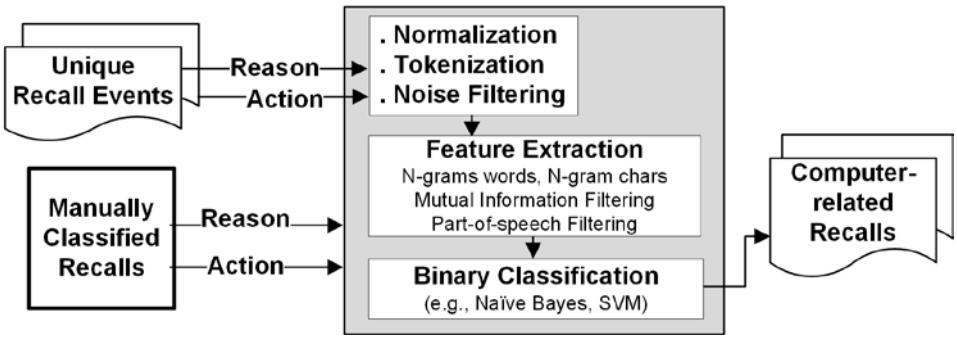


Figure 2.7. Analysis steps for identifying computer-related recalls.

transforms the filtered text in the *Reason for Recall* (or *Action*) field into a term vector of features. Then it uses the set of manually classified computer-related recalls from the study presented in Section 2.3, as a training set to create a binary classifier that distinguishes the computer-related recalls from non-computer-related recalls.

Using a bad-of-words model [31], MedSafe transforms each recall record to a sparse vector representing the occurrence counts of keywords (from a pre-defined dictionary) in the field. We define this vector (or dictionary of keywords) by extracting N-gram words or N-gram characters (of different lengths) from the *Reason for Recall* field and further filter the dictionary using part-of-speech (POS) tagging (e.g., to only keep nouns, adjectives, and verbs) or mutual information metrics to only keep the highly relevant features.

Using a mutual information (MI) metric [31] we measure how much the presence or absence of a keyword in the *Reason for Recall* field of a recall event contributes to the classification of that recall into the computer-related class. For each keyword *k*, the *MI* metric is calculated using the following equation:

$$MI(R_k, C_r) = \sum_{i,j \in (0,1)} P(R_k = i, C_r = j) \log_2 \frac{P(R_k = i, C_r = j)}{P(R_k = i).P(C_r = j)} \tag{2.3}$$

where the R_k and C_r are binary indicator random variables, respectively representing whether the recall event *R* contains the keyword *k* ($R_k = 1$), and whether the recall event *R* is in the computer-related class ($C_r = 1$). If we let $N(R)$ be the total number of recalls in the training set, $N(R_k = i)$ the number of recalls in the training set that contain ($N(R_k = 1)$) or do not contain ($N(R_k = 0)$) keyword *k*; $N(C_r = j)$ the number of recalls in the training set that that are computer-related ($N(C_r = 1)$) or not computer-related ($N(C_r = 0)$); and $N(R_k = i, C_r = j)$ the number of recalls that are at the intersections of these sets, then each of the probabilities in Equation (2.3) can be calculated using the maximum likelihood estimates (MLE). For example:

$$P(R_k = i, C_r = j) = \frac{N(R_k = i, C_r = j)}{N(R)} \tag{2.4}$$

The keywords are then sorted according to their *MI* scores and the top half of the list is used as the dictionary for feature extraction.

For the classification of recalls, we used multinomial Naïve Bayes and linear Support Vector Machine (SVM) [31] algorithms. The Naïve Bayes classifier calculates the probability of a recall *R* being a computer-related recall versus probability of *R* being a non-computer-related recall and assigns *R* to the class with the higher probability, as follows:

$$Class(R) \propto \underset{c \in \{0,1\}}{argmax} \left[log\, P(c) + \sum_{1 \le i \le M} logP(k_i|c) \right] \tag{2.5}$$

where $P(c)$ is the prior probability of a recall being in computer- (c = 1) or non-computer-related (c = 0) class, and is calculated based on the training data. $P(k_i|c)$ is the conditional probability of keyword k_i appearing in a computer-related recall and *M* is the total number of keywords in the vocabulary (extracted in the feature selection phase).

The SVM classifier calculates the dot product between a recall feature vector (occurrence counts of keywords in the recall) and a weights vector (that is learned based on the training data) and if

Table 2.6. Performance of different classifiers in identifying computer-related recalls.

Feature Set	Classifier	Precision	Recall	F1-Score
"Reason for recall" Unigram words Part of speech filter (NN, ADJ, V) Mutual information filter	**Naïve Bayes**	0.69	0.87	0.77
"Reason for recall" Unigram words	**SVM**	0.69	0.85	0.76
"Reason for recall" N-gram words (N = 1, 2, or 6)		0.73	0.84	0.78
"Reason for recall" + "Action" Unigrams, Bi-grams, 6-gram chars		0.79	0.85	0.82

the sign of the result is positive, declares a computer-related recall. The dot product combination defines a decision boundary with the maximum distance to the two classes. The SVM model parameters (weights vector) are learned using a stochastic gradient descent (SGD) optimization method that minimizes the training error calculated based on a hinge loss function [32].

Evaluation: We evaluated different combinations of feature sets and classification methods as shown in Table 2.6. We used Python NLTK library [19] and MeTA (an open-source toolkit for Modern Text Analysis) [32], [33] to implement and test the Naïve Bayes and SVM classifiers.

In order to assess the performance of each classifiers, we performed a ten-fold cross-validation, using 4,398 recall events from 2007–2011, which were manually labeled in our previous study (Section 2.3). Each subset of 439 recalls was used as the test data, and the remaining (3,959) recalls were used for feature selection and training. The standard metrics, including *sensitivity*, *specificity*, and *F-Score* (considers both *precision* and *recall* metrics with a value of 100% indicating perfect results) were used for evaluation of the results and were averaged over the ten runs of experiments. MedSafe achieved an average sensitivity and specificity of 88.2% and F-Score of 77.9% using Naïve Bayes classifier and both POS and MI filters on unigram words. As shown in Table 2.6, the F1-score only improved by 1% using the SVM classifier. However, when we added both the *Reason for Recall* and *Action* fields to the feature set, the F1-socre of the SVM classifier improved by 5%. These results show that potentially there is important information in the *Action* fields of the recalls that distinguishes the computer-related recalls from non-computer-related recalls.

We also evaluated the recall classification using basic Naïve Bayes on the new (unseen) data. We used the manually labeled data from 2007–2011 as a training set to classify the recalls submitted for the years 2012 and 2013. Our experiments showed that of 5,011 recall records submitted during 2012–2013, 2,466 (49.2%) were unique recall events, from which 634 (25.7%) were computer-related. By manual review of these results, we found that MedSafe identified the computer-related recalls with a sensitivity of 93.8% and a specificity of 95.8% (Precision: 88.7%, Recall: 93.8%, and F1-Score: 91.1%).

2.4.4. Device Category Classification

MedSafe integrates the device category information, including the *Medical Specialty*, *Product Code*, *Device Name*, and *Submission Type* with the information in the recall records. It first extracts device classification data from the FDA website and then for each *Product Code* (or device category) opens the corresponding online TPLC record and parses its HTML file to get the list of all related recall records in that device category. For about 16% of the recalls, the device classification information was not available through TPLC database. For the computer-related recalls that could not be classified using *Recall Number*, MedSafe identifies the device category information from the recalls with the most similar *Product Name* and *Manufacturer* information, which also have the device category information are available.

After the publication of this work, the FDA online database has started to provide a link to the TPLC records of recalled devices in their recall records, which can be used as a more efficient way

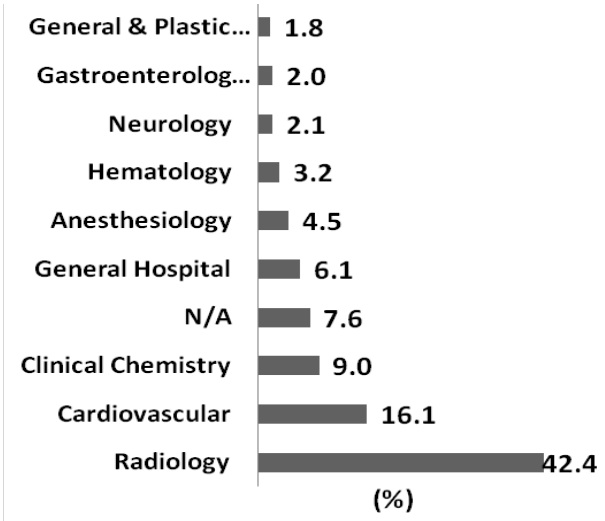


Figure 2.8. Computer-related recalls across different device categories.

for extraction of the device category information. Figure 2.8 shows the medical device categories with the highest number of computer-related recalls reported during 2007–2013. About 58% of the computer recalls were related to Radiology and Cardiovascular devices, such as automated external defibrillators and linear accelerators.

2.5. Related Work

Although many previous works from dependability community focus on failure data analysis of different computing systems such as high-performance computing systems [34], Windows NT-based computers [35], mobile phones [36], telephony systems [37], [38], and bluetooth personal area networks [39], no attention had been paid to the analysis of failures in medical devices that are built using computers. The only available data on failures and safety issues of medical devices are the reports collected by the Food and Drug Administration (FDA) databases.

In the biomedical and clinical engineering community there are several works on the failure analysis of medical devices based on the studying adverse events reports and recalls. However, most of these works are only focused on small subsets of recalls and adverse events data to derive statistics on the number of software- [4] – [7], computer- [40] – [42], and security- [6], [9] related reports. Among these works, only [4] presented an extensive analysis on the root causes of failures and their symptoms in software medical devices. There are also several studies on recalls and adverse events reported for specific types of devices such as cardiac pacemakers and implantable defibrillators (ICDs) [8], [43] – [46]. Most of these works used either keyword search or manual reviewing of the recalls and adverse event descriptions, which requires a significant amount of human effort and still may not produce accurate results due to human mistakes or inadequate list of keywords.

To the best of our knowledge, the approach presented in this dissertation is the first effort toward identifying different kinds of failures in computer-operated devices and studying the relation between the recalls and adverse events reports by making associations among the causes of failures, their symptoms, and measuring the impact of failures on patients and manufacturers in terms of severity of events, number of devices on the market that were affected by the recalls, and the cost of device repairs and removals.

2.6. Conclusions

By learning from past incidents we can better identify the potential hazards, safety requirements, and required risk mitigations for designing the next generation of devices and prevent reoccurrence of similar adverse events in the future. Medical device incidents are typically reported to the FDA by the users, manufacturers, and investigators. A major component of these reports are human-written narratives describing adverse events, reasons for recalling a device, or corrective actions taken by the manufacturers. Automated analysis of these reports is a challenging task, requiring semantic interpretation of natural language text. We developed *MedSafe*, a toolset for large-scale analysis of recalls and adverse event reports from the public FDA databases [3]. By using techniques from natural language processing, information retrieval, and machine learning, *MedSafe* enables deeper understanding of causes of device failures and statistically confident measures on their impacts.

CHAPTER 3
SYSTEM-THEORETIC ANALYSIS OF INCIDENTS

3.1. Overview

Analysis of adverse events data provides valuable insights on possible system hazards and can be used to assist enhancement of system safety in order to prevent future incidents caused by malfunctioning medical devices. However, traditional accident modeling and analysis techniques (e.g., *root cause analysis* commonly used in medicine) tend to focus on direct causal relationships between events leading to an accident (based on chains of events or Swiss cheese models [51]) and often conclude with subjective emphasis on a single (root) cause, such as physical device failures or human errors. Other causal and contextual factors and underlying conditions, that indirectly contribute to adverse events, are not thoroughly considered in such analyses; e.g., unsafe interactions among system components and human operators; inaccurate operators' mental models of the system; or the circumstances (system states) under which the actions are taken or decisions are made [12]. Therefore, those approaches are often insufficient for understanding all the causes contributing to accidents in today's complex medical systems [52] – [54] and recommendations made based on such analyses often fail to enhance safety and similar accidents recur.

System-theoretic accident causality models such as STAMP (Systems-Theoretic Accident Model and Processes) [12], overcome this limitation by modeling accidents as complex dynamic processes resulting from inadequate control mechanisms that violate safety constraints. Driven by concepts in systems and control theories, STAMP provides a modeling framework for hazard analysis and accident investigation and is applied in several safety-critical domains such as aviation [55], medical devices [56], [57], automotive [58], [59], and transportation [60], [61].

We apply the STAMP modeling framework to analysis of adverse events in safety-critical medical devices. To facilitate the system-theoretic analysis of large sets of adverse event reports, we propose a new ontology model based on human-in-the-loop control structures to guide the

---

This chapter contains material from the published works [47] – [50], coauthored with Z. Kalbarczyk, R. K. Iyer, J. Raman, N. Leveson, T. Kesavadas, and S. Small.

automated semantic analysis of the adverse events. Specifically, the ontology model defines the device-specific entities and relations that correspond to the key safety-related information (e.g., contextual factors, unsafe operator actions, device malfunctions, and patient status) needed in causality analysis based on STAMP and should be extracted from the unstructured narratives of the reports.

We demonstrate the effectiveness of this approach in a case study of adverse events reported for robotic systems used in minimally invasive surgery. We analyzed the FDA data on deaths, injuries, and malfunctions that occurred during robotic procedures and characterized:

- ***System hazards and potential causal factors*** that led to unsafe control actions during the procedures and caused accidents.
- ***Potentially inadequate safety mechanisms*** in both system design and operational practices in robotic surgery that led to health-threatening events, including patient injuries and death.

Section 3.2 provides an overview on the common challenges faced in analysis of reports from the FDA MAUDE database, Section 3.3 presents background on system-theoretic accident modeling and analysis using STAMP, exemplified by our analysis of sample adverse events in robotic surgery. In Section 3.4, we describe our proposed ontology model for system-theoretic semantic analysis of adverse event reports. Finally, Section 3.5 presents our case study of analyzing the adverse events related to da Vinci surgical system [13], the only FDA approved robotic device for minimally invasive surgery.

3.2. Challenges in Analysis of FDA Adverse Event Reports

Figure 3.1 shows an example MAUDE report on an adverse event occurred during a robotic cardiothoracic procedure [62] (for more detailed description on the structure of adverse event reports, refer to Section 2.2.2). There are several challenges (even for the FDA analysts [63]) in analyzing the FDA adverse event reports:

(i) According to the FDA [64] and several studies in the literature [9], [65], [66], the MAUDE database may not provide an accurate representation of true rates and severity of adverse events due to underreporting as well as inaccuracy, inconsistency, and duplication in the reports that are submitted by the volunteer reporters.

INTUITIVE SURGICAL, INC. DA VINCI S SURGICAL SYSTEM ENDOSCOPIC INSTRUMENT CONTROL SYSTEM Back to Search Results

Model Number IS2000 A5.1P4
Event Date 07/17/2008
Event Type Injury
Event Description

This report is being filed based an isi investigation concluding 08/08/2008. (b)(4) was received on 08/01/2008 from the (b)(6). It was reported that during da vinci s bilateral internal mammary arteries revascularization procedure, the customer experienced a system error code #23. With the assistance of an isi (b)(4), the site powered down the system to clear the fault. The site continued with the procedure, however, the system error reoccurred. The site disabled the endoscopic camera manipulator (ecm) to continue the case. The site then elected to manually manipulate the camera and endoscope for approximately 5 to 6 hours when a loss of carbon dioxide insufflation occurred resulting in the heart pushing up into the endoscope two times causing lacerations to the patient's right ventricle. The procedure was converted to a thoracotomy to performed a non robotic repair of the damaged ventricle with several stitches and to complete the planned procedure. After the 14 hour procedure, the patient could not be extubated necessitating a tracheostomy. As of (b)(6), the patient remained under care at (b)(6).

Manufacturer Narrative

The investigation conducted by (b)(4) concluded that the system error code #23 experienced by the customer was associated with a configured embedded sterilizer setup joint (cfg, essj) and remote arm controller (rac). The embedded sterilizer for setup-joint is the printed circuit assembly (pca) inside a system arm that monitors the potentiometer for each of four joints and their associated backup potentiometers. The rac consists of five printed circuit assembly boards which operate together to provide control of the system arms. The system was repaired by replacing the affected cfg, essj and rac. System error code #23 is reported by software to denote that the hardware wheel "wdog" has tripped on one of the digital communication links in the system. This means that the system cannot reliably communicate over the digital link and therefore cannot continue normal operation. Communication faults occasionally occur due to typically either faulty electronics or poor connections in the communication link. The error 23 fault indicated by the system in this case pointed to a communication error involving the ecm's rac and essj modules. The system was repaired as the (b)(4) both removed the electronics that could have experienced an intermittent failure and re-secured all of the connections involved. Field experience has shown that these measures are effective in resolving this type of system communication issue. As of (b)(6) 2007, the site has continued to use the system and has not reported recurrences of this issue.

Figure 3.1. A sample adverse event report from the FDA MAUDE database, available at [62].

(ii) The fields such as *Device Name*, *Product Code*, are often inconsistent across multiple reports related to the same device, and *Device Problem* fields are either missing or represent incorrect information about the actual problems encountered by device operators.

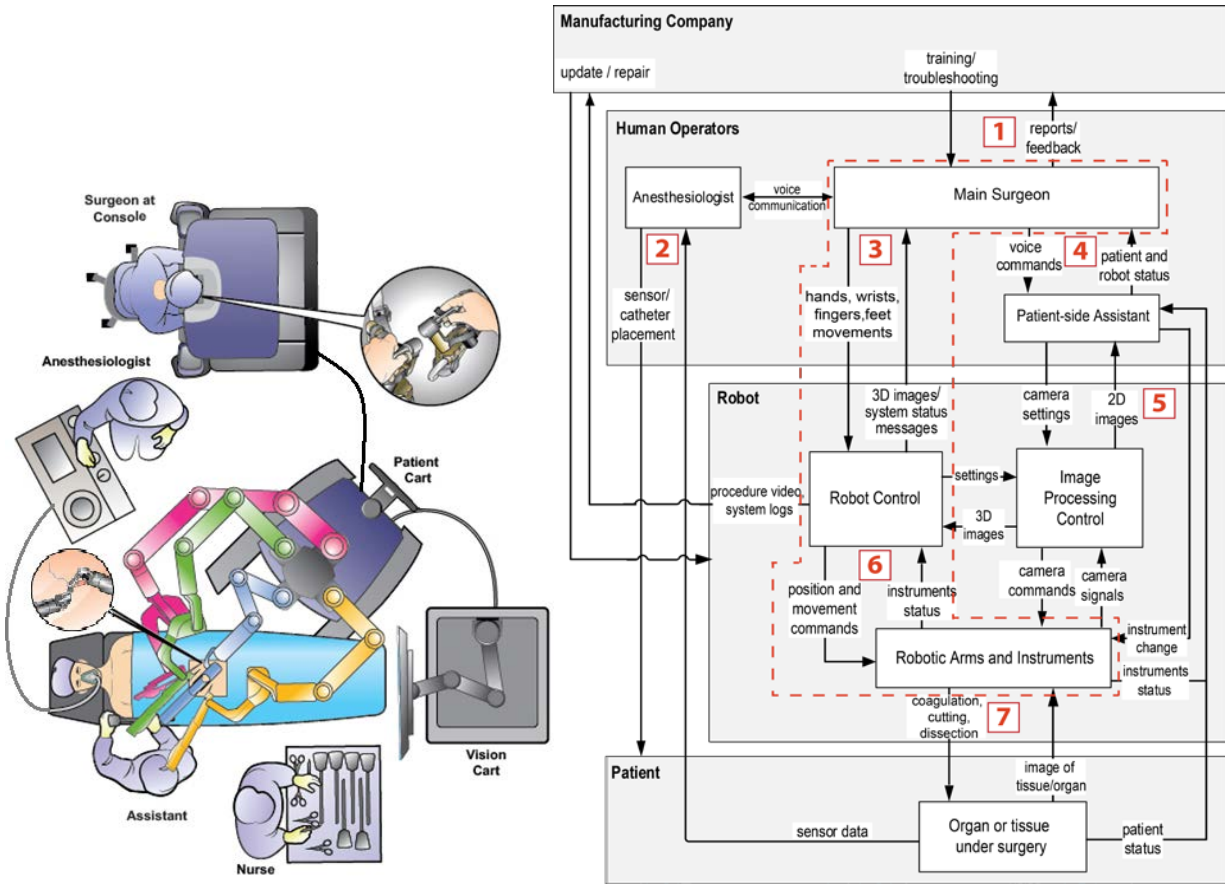
(iii) The *Event Description* and *Narrative* fields of the reports, which provide the most important information on the event and patient impacts, are written in free-form natural language text and are often very abstract, ambiguous, conflicting to each other, and difficult to analyze without considering the information provided in the other fields and understanding the underlying factors involved in the incidents.

However, the reported deaths, injuries, and device malfunctions provided by the MAUDE are valuable data if treated as a *sample set* to estimate the *prevalence of adverse events* and identify *examples* of their major causes and patient impacts (see Appendix A for more details on the problem of underreporting). Although an external retrospective review of these reports cannot determine all the causes involved in incidents, it can help to identify example hazards that were not detected and mitigated during hazard analysis and design processes and to facilitate design of safety controls that can mitigate those hazards and prevent similar incidents in the future.

3.3. System–Theoretic Analysis of Adverse Events

In STAMP the systems are modeled as hierarchical control structures, where the components at each level of the hierarchy impose safety constraints on the activity of the levels below, and communicate their conditions and behavior to the upper levels. The layers of the control structure could span from the physical components to human operators, up to higher levels in manufacturing, management, and regulation. Accidents are considered as complex dynamic processes resulting from violation of safety constraints at different layers of control structure [12].

Figure 3.2(a) shows the typical system setup in a robotic procedure, including human operators (surgeon, surgical assistant, nurse, and anesthesiologist), surgical robot (surgeon console, vision cart, and patient cart), and the patient. We modeled the hierarchical control structure of a surgical robot as shown in Figure 3.2(b), based on our review of publicly available documents on commercial and open-source robotic surgical platforms including da Vinci Surgical System [13].



(a) (b)

Figure 3.2. (a) A typical setup of robotic surgical systems for minimally invasive surgery (Adapted from the Intuitive Surgical, Inc. [13]), (b) Hierarchical control structure of the robotic surgical system.

In STAMP, the interactions among system components and operators are modeled as control loops (e.g., loop 3) composed of the actions or commands (e.g., hand and foot movements) that a controller (e.g., main surgeon) takes/sends to a controlled process (e.g., robot control) and the response or feedback (e.g., images/status messages on surgeon's console) that the controller receives from the controlled process.

Every controller in the system uses an algorithm to generate the control actions based on a model of the current state of the process that it is controlling. For example, control loops 3 and 6 (outlined by dashed lines in Figure 3.2(b)) are further refined in Figure 3.3 to illustrate details on

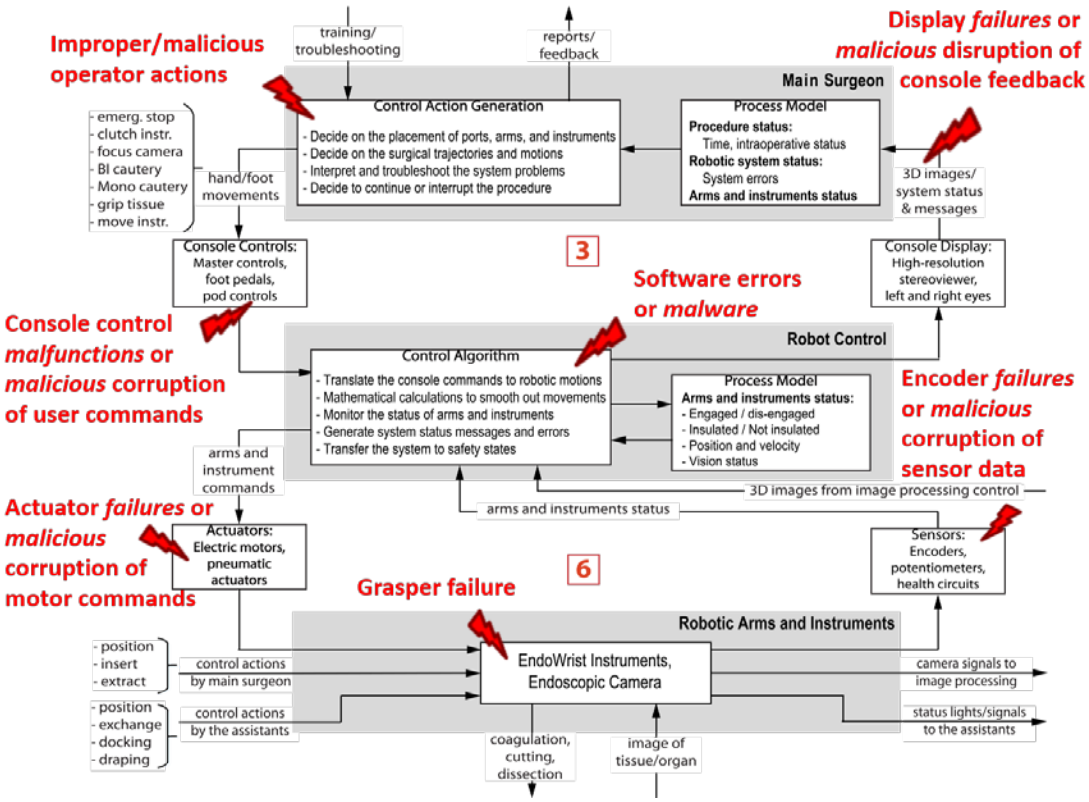


Figure 3.3. Control actions and process models for control loops 3 and 6 (highlighted in Figure 3.2(b)), Example causal factors are marked with ⚡.

the interactions among the main surgeon, robot control, and robotic arms/instruments. The control action generation, process model, examples of control action, and components that enable the action and feedback between the main surgeon and the robot are highlighted here.

In every control loop, the control actions taken (e.g., clutch an instrument) by the controller (e.g., surgeon) change the state of the controlled process (e.g., the instrument will be engaged). The feedback (e.g., messages on the console) sent back from the controlled process (e.g., robot

Table 3.1. Accidents and system hazards in robotic surgery.

Accidents
A-1. Patient expires during or after the procedure.
A-2. Patient is injured during procedure or experiences serious complications after the procedure.
A-3. Surgical personnel are injured by the surgical system.
A-4. Surgical system or instruments are damaged or lost.
System Hazards (Possible accidents)
H-1. Robot arms/instruments move or cut or apply energy to an unintended location, of an unintended amount, or at an unintended time.
H1-1. Right location, right amount, wrong time
H1-2. Right location, wrong amount, right time
H1-3. Right location, wrong amount, wrong time
H1-4. Wrong location
H-2. Robotic system, arms, or instruments are subjected to collision or unintended stress.
H-3. Robotic system becomes unavailable or unresponsive during procedure.

control) updates the process model used (e.g., surgeon's mental model) by the controller (e.g., surgeon observes on the console that the instrument is registered). Any accidental or malicious flaws or inadequacies in the algorithm, process model, or feedback used by a controller could possibly lead to unsafe control actions and hazardous states in the system.

Causal analysis of accidents using STAMP (called CAST) [12], starts with identifying the system hazards and violated safety constraints involved in the accident. Then the operation of components and their interactions in each control loop of the control structure are examined to identify potential causal factors for the safety hazards, including improper operations of components, inadequate interactions (control actions or feedbacks) among them, inaccurate models used by the controllers, and any contextual factors that may have contributed to safety hazards (marked with in Figure 3.3).

Based on the manual review of almost 1,500 adverse event reports and knowledge of robotic surgical systems functionality, we classified the robotic surgery accidents into the following four types: patient deaths (A-1), patient injuries during the procedure or serious complications experienced after the procedure (A-2), surgical personnel injuries caused by the system (A-3), and costly damage to the surgical system or instruments (A-4). We also identified three main types of hazards or set of unsafe system states that could lead to those accidents (see Table 3.1).

We then reviewed the textual description of more than 1,500 injury and death events (A-1 and A-2 accident types) and selected a subset of reports that included relatively more detailed event descriptions. Then based on the description of each report, we examined the characteristics and

responsibilities of each of the controllers in the safety control structure (Figure 3.2(b)), extracted the information related to flawed control actions and unsafe interactions in the system that contributed to the adverse events, and identified examples of potential causes and underlying context for them. The majority of the injury and death reports provided little or no detailed information about the possible causes of the adverse events or only reported on the single causal factors (such as inherent risks of surgery, human errors, and device malfunctions) involved. However, a careful review of injury and death reports helped us to identify several reports that included information on multiple causal and contextual factors contributing to the adverse events.

For example, Table 3.2 shows the description of an adverse event reported in 2008 (MAUDE report no. 2240665, also shown in Figure 3.1), where an electronic component failure, a non-recoverable system error, inadequate troubleshooting of technical problems, and possibly ineffective decisions made by the human operators, each played a role in a very long procedure time and consequently, a patient injury. According to the manufacturer narrative, a *chain-of-events*-based analysis concluded that the root cause of the event was an electronic component failure and the recovery actions taken based on such conclusion were to repair the faulty components of the system. But a systematic CAST analysis provides us with a different view on other multidimensional causal factors that were also involved in the event.

As shown in Table 3.2, we identified the system hazards (e.g., robotic arms/instruments move/cut/apply energy to an unintended location), the potentially unsafe control actions taken in each control loop (e.g., surgeon deciding to manually operate the endoscopic camera for a long period of time, in control loop 3), possible causes for those unsafe actions (e.g., electronic component failure preventing the automatic manipulation of camera or lack of detailed troubleshooting procedures), and the context in which those decisions were made (e.g., the type of procedure and the time spent before deciding to convert the procedure). The important pieces of information used in the analysis are underlined in the textual description of event and the potentially flawed controls and feedbacks in different loops of control structure are highlighted in Table 3.2. Table 3.3 shows example events from the MAUDE database in which similar kinds of control flaws in different control loops of the control structure led to hazards and patient injuries. Examples of common flaws included limited training of surgical team (loop 1), inadequate troubleshooting of technical problems (loop 3), inadequate interactions between surgeon and robot at the console (loop 3), and device and instrument malfunctions (loops 6 and 7).

Table 3.2. Causal analysis of an example adverse event report (report no. 2240665) using STAMP.

Event Date: 07/17/2008	**Event Type:** Injury	**Patient Outcome:** Required Intervention, Life Threatening

Event Description: It was reported that during da vinci s bilateral internal mammary arteries revascularization procedure, the customer experienced a system error code #23. With the assistance of the company representative the site powered down the system to clear the fault. The site continued with the procedure, however, the system error reoccurred. The site disabled the endoscopic camera manipulator (ecm) to continue the case. The site then elected to manually manipulate the camera and endoscope for approximately 5 to 6 hours when a loss of carbon dioxide insufflation occurred resulting in the heart pushing up into the endoscope two times causing lacerations to the patient's right ventricle. The procedure was converted to a thoracotomy to perform a non robotic repair of the damaged ventricle with several stitches and to complete the planned procedure. After the 14 hour procedure, the patient could not be extubated necessitating a tracheostomy.

Manufacturer Narrative: The investigation concluded that the system error code #23 experienced by the customer was associated with a configured embedded sterilizer setup joint (cfg, essj) and remote arm controller (rac). The embedded sterilizer for setup-joint is the printed circuit assembly (pca) inside a system arm that monitors the potentiometer for each of four joints and their associated backup potentiometers. The rac consists of five printed circuit assembly boards which operate together to provide control of the system arms. The system was repaired by replacing the affected cfg, essj and rac. System error code #23 is reported by software to denote that the hardware wheel "wdog" has tripped on one of the digital communication links in the system. This means that the system cannot reliably communicate over the digital link and therefore cannot continue normal operation. Communication faults occasionally occur due to typically either faulty electronics or poor connections in the communication link. The error 23 fault indicated by the system in this case pointed to a communication error involving the ecm's rac and essj modules. The system was repaired as the electronics that could have experienced an intermittent failure were removed and all of the connections involved were re-secured.

Causal Analysis using STAMP

Safety Hazards:

H-1. Robot arms/instruments move/cut/apply energy to an unintended location:

– Heart pushed up to the endoscope two times.

H-3. Robotic system becomes unavailable or unresponsive during procedure:

– System error was not cleared after system reset.

Unsafe Control and Causal Factors:

Loop 1:

– Further support on resolving the system error not provided by company representative.

– Inadequate consulting with the company about manual manipulation of endoscopic camera.

Loop 3:

– Unsafe decision made by surgeon to manually operate the endoscopic camera for long period of time.

– Unsafe action taken by surgeon to continue the procedure with low level of carbon dioxide insufflation.

– Lack of detailed feedback from robotic system on the error and best troubleshooting/recovery actions.

Loop 4:

– Inadequate feedback/information provided by assistant to surgeon on the patient insufflation status.

Loop 6:

– Electronic component failure prevented the robotic manipulation of the endoscopic camera

Contextual Factors:

Cardiothoracic surgery, long procedure time (> 6 hours)

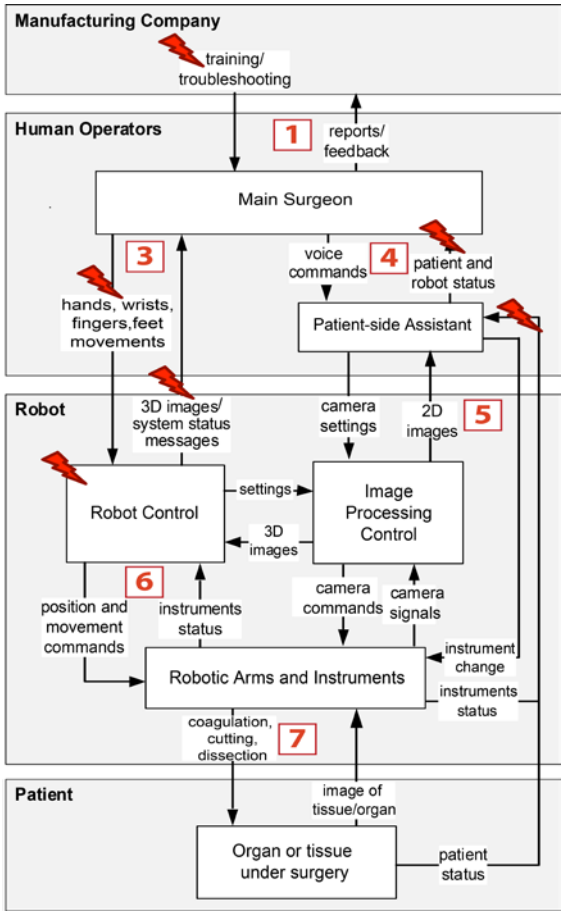


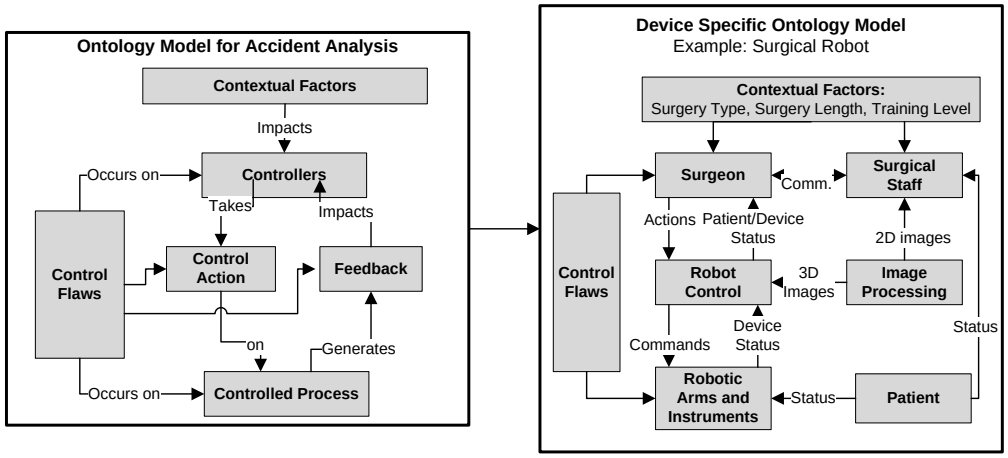
Table 3.3. Accidents, system hazards, and potential control flaws involved in example adverse events (The full description of the adverse event reports shown here is available in Appendix B).

MAUDE Report No. (Year)	Potential Control Flaws (Unsafe Control Actions, Feedbacks, Processes)	Control Loops	Contextual Factors	Hazards	Accident
2567858 (2012)	- Inadequate training of surgeon or surgical staff by company - Inadequate action (instrument change) by surgeon assistant - Lack of feedback from system or instrument to assistant	(1) (5) (5)	45 minutes into procedure, Converted to open surgery, Patient stable for one day	H1	A1 A2
1891889 (2010)	- Inadequate feedback (vision) from robot to surgeon	(3)	Hysterectomy procedure, Ureteral injury	H3	A2
1760256 (2010)	- Unsafe decision/action by surgeon to continue surgery in 2D - Loss of feedback (signal) from robot camera (broken cable)	(3) (8)	Prostatectomy procedure, Procedure lasted 8 hours, Leg injury after procedure	H3	A2
2494890* (2012)	- Unsafe action by surgeon (used incorrect electro-cautery)	(3)	Cardiac (CABG) procedure, Small diaphragm burn	H1	A2
2476271* (2012)	- Inadequate feedback on electro-cautery connection/status - Incorrect action by assistant (wrong instrument connection)	(3) (5)	Hysterectomy procedure, Bowel damage		
3024317* (2013)	–Inadequate interface or design of electro-cautery device	(4)	Prostatectomy Bowel injury		
2632716 (2012)	- Lack of status of tip cover accessory to surgical team - Faulty instrument (arcing due to insulation failure) - Inadequate training or procedures for handling instruments	(6) (7) (1)	Hysterectomy procedure, Vascular injury, Converted to open surgery	H2	A2
3473388 (2013)	- Unsafe action by surgeon (incorrect port placement) - Inadequate feedback (instrument status) to surgeon - Inadequate training of surgeon or surgical staff - Unsafe action (over-rotation) of the (mtm) instrument	(3) (6) (1) (7)	Hysterectomy procedure, Artery nicked, Converted to open surgery	H1 H2	A2 A4

The manual causal analysis of thousands of adverse event reports needs significant human effort and still cannot provide a comprehensive understanding on all the causal factors and statistically significant measures of their importance. In Section 3.4, we present an approach that leverages the STAMP accident causality model to structure the unstructured data on adverse events and facilitates the system-theoretic accident causality analysis process.

3.4. Semantic Parsing of Adverse Event Reports

We propose a novel ontology model based on the hierarchical control structures used for modeling human-in-the-loop systems in the STAMP causality framework (as shown in Figure 3.4). The proposed ontology is specified based on the control structure of the target medical device during the design or validation phase. The entities and relations in the ontology model correspond to the components and interactions between them in the system control structure and define the



(a) (b)

Figure 3.4. Control-structure based ontology model for causal analysis of accidents: (a) Generic ontology model, (b) Device-specific ontology model for a robotic surgical system.

safety information that should be extracted from the reports. More specifically, the device-specific ontology model is populated with pre-defined sets of dictionaries and syntactic rules that can be used to identify appearance of ontology entities and relations in the natural language text. This ontology model can be used by semantic parsing tools to annotate the entities and relations in the adverse event descriptions.

Table 3.4 shows different classes in the generic ontology model (Figure 3.4(b)), which are defined based on the entities in the system control structure (controllers and controlled processes), their interactions (control action and feedback), and the contextual factors and control flaws considered in the causal analysis using STAMP, including: (i) improper operation of any components in the control loops, (ii) inadequate, ineffective, or missing control actions provided by the controllers, and (iii) inaccurate or missing feedbacks sent from the controlled processes. Each class is further divided into subclasses, specific to the medical device under investigation, which here is a robotic surgical system (shown in Figure 3.4(b)).

The last two columns of Table 3.4 show the example tags and relevant keywords as well as the syntactic rules for extracting and annotating the entities of each sub-class in the text. Here the dictionaries are created based on the domain knowledge and the online documents describing the target system functionality and parts. The syntactic rules are extracted through an iterative process, wherein we manually reviewed random subsets of the reports, looked for the patterns in keywords, and continually updated the relevant keywords by assessing the searching results.

Table 3.4. Classes and sub-classes in the ontology model for accident analysis of surgical robots.

Class	Sub-Classes	Related Tags and Keywords	Example Syntactic Rules
Contextual Factor	Surgery Class	CLASS: gynecologic, urologic, cardiothoracic, colorectal TYPE: hysterectomy, prostatectomy, mitral valve repair	Any TEMPORAL terms Preceded or succeeded by NUM - TIME_UNIT pattern
	Temporal Information	TEMPORAL: during, before, after, into, prior, for, past NUM: one, two, three, four, half, an, few, several, 1, 10, 20 TIME_UNIT: hour*, hr*, minute*, min*	
Controller	Surgical Staff	STAFF: surgeon, staff, site, nurse, physician, anesthesiologist	
	Robot Control	DEVICE: console, pedal, system, device, robot, controller	
Controlled Process	Robot Control	DEVICE: console, pedal, system, device, robot, controller	Any DEVICE or INST (as object)
	Robotic Arms/ Instruments	INST_GENERAL: arm, instrument, object, component, part INST_NAMES: endoscope, cannula, manipulator, camera	
	Patient	PT: patient, pt, subject	
Control Action	Decision	chose, select, decide	Any STAFF (as subject) succeeded by a VERB
	Troubleshoot	troubleshoot, convert, reschedule, abort, reset, reboot, restart, shut off/down, power off/down, turn off	
	Actuation	move, manipulate, cut, retrieve, remove	
	Communication	call, ask, contact	
Feedback	Display	VIDEO: surgeon console, assistant monitor, touch screen, vision, video, image, camera	Any STAFF (as subject) succeeded by a VERB
	Other	FEED: observe, notice, seen, found, experienced, saw	
Control Flaw	Controller	ERR: mistake, error, incorrect, non-intuitive, wrong, sudden, unintended, unexpected. erratic, uncontrolled, PART: pieces, fragments, shredding, burn, hole NEG: not, stopped OP: work, recognize, rotate, move, open, close, grasp, manner, etc. COND: arc, charr, spark, fell, broke, drop, snap [into/in/off/inside]	-DEVICE/STAFF succeeded by ERR -INST_GENERAL/NAMES preceded by PART, succeeded by POST -INST_GENERAL/NAMES succeeded by NEG/ERR-OP or COND
	Controlled Process		
	Control Action		
	Feedback	ERR: no, loss, lost, inaccurate, incorrect, error, problem, issue COND: double, black, blurry, foggy, not aligned	VIDEO or FEED preceded or succeeded by ERR of COND or PRE
Accident Type	Staff/Patient Injury	INJ: injury, tear, burn, puncture, perforation, laceration, organ damage, bleeding, avulsion, rupture, necrosis, harm	-INJ terms preceded or succeeded by PT or STAFF
	Patient Death	death, expire	-INJ terms preceded by PT

As discussed in Section 3.5, after extracting the data, filtering non-relevant reports, and coalescing duplicate reports, MedSafe uses part-of-speech and negation taggers along with the semantic dictionaries and syntactic rules provided in Table 3.4 to mark up different patterns related to the concepts in the ontology model. Table 3.5 shows snippets of three example reports along with the semantic annotations by the MedSafe. *The first example* shows a device malfunction (instrument breakage) occurred during a hysterectomy procedure, which was addressed by a control action. *The second example* shows an display malfunction (image in the stereo viewer going back) which occurred 30 minutes into the procedure. Here, a patient injury was identified

by the semantic tagger, but the negation tagger detected the negative sentiment of the sentence and this markup was invalidated. *The third example* shows an instance of a system error that is propagated as a flawed feedback to the surgical team (site). The troubleshooting control actions taken by the team such as communication with the company, powering off, and restarting the system are highlighted by the semantic tagger.

Table 3.5. Example reports annotated by the semantic tags (highlights in bold) and extracted patterns (highlighted by arcs).

The patient was undergoing a robotic total laparoscopic **hysterectomy\SurgeryType** with bilateral salpingectomy. The wires on the instrument\INST broke\COND.The fragments\INST were retrieved\ControlAction from the patient's abdomen. Flawed Controlled Process (Device malfunction)
It was reported that approximately **30 minutes into\Temporal_Information** a partial **nephrectomy\SurgeryType** procedure, the right eye **image\VIDEO** in the high resolution stereo viewer went **black\COND**. Flawed Feedback (Display) Troubleshooting (convert) Unable to resolve the issue, the **surgeon\STAFF** decided to **convert\ControlAction** to traditional open surgical techniques to complete the planned procedure. **No\NEGATION patient\PT harm\INJ** was reported. Invalidates the markup → ~~Accident Type (Patient Injury)~~
During a surgical procedure, the **site experienced\Feedback** multiple instances of **system\DEVICE error\ERR** code 23021. Flawed Controlled Process (Device malfunction) The **site\Staff contacted\ControlAction** isi for technical support engineering (tse) assistance. Communication Troubleshooting (restart) With the assistance of the tse the **site\STAFF** performed an emergency **power off\ControlAction** the **surgeon side cart (ssc)\INST** and the **patient side cart (psc)\INST**and **restarted\ControlAction** the system; however, the issue persisted. Troubleshooting (restart)

The semantic labeling of the reports can be done more systematically by employing formal language models and automatic semantic role labeling techniques [67], [68] as well as more advanced natural language processing and knowledge discovery techniques (e.g., para-phrasing [69], topic modeling [70], and unsupervised learning methods [71]) to automatically infer and extend the set of dictionary keywords and syntactic rules based on the seed examples which are created manually.

The initial implementation of the proposed technique was done for labeling the contextual factors (e.g., surgery classes and types, temporal information), accident types (e.g., patient injuries and complications), and a subset of control flaws (including device and instrument malfunctions and troubleshooting control actions) as part of analysis of adverse events for robotic surgical systems (shaded sub-classes in Table 3.4). The accuracy of annotations was evaluated by manual

review of the reports. The implementation of the rules for labeling all the classes in the ontology model and design of experiments for more comprehensive evaluation of the results is the subject of future work.

3.5. Analysis of Adverse Events in Robotic Surgery

Use of robotic systems for minimally invasive surgery has rapidly increased during the last decade. Between 2000 and 2013, over 1.75 million robotic procedures were performed in the United States across various surgical specialties [72]. Surgical robots enable conducting complex minimally invasive procedures with better visualization, increased precision, and enhanced dexterity compared to laparoscopy. Robotic devices provide 3D magnified views of the surgical field and translate the surgeon's hand, wrist, and finger movements into precisely engineered movements of miniaturized surgical instruments inside patient's body. The Intuitive Surgical's da Vinci robot [13] is currently the only surgical robot approved by the U.S. Food and Drug Administration (FDA), for performing various types of procedures in urologic, gynecologic, general, cardiothoracic, and head and neck surgery. There are also other robotic systems designed for minimally invasive surgery in areas such as neurosurgery and orthopedic surgery (e.g., MAKO Surgical's RIO Robotic Arm Interactive System for orthopedic surgery [73]) or for research in teleoperated robotic surgery (e.g., the da Vinci research kit [74] and the RAVEN II surgical robot [14], [15]).

This study focuses on assessing the safety and effectiveness of robotic surgical systems used in minimally invasive surgery, by analyzing safety incidents experienced during robotic procedures. We retrieved all the nationwide adverse event reports collected by the publicly available FDA MAUDE database [64] over the 14-year period of 2000–2013. Our analysis included *estimating the prevalence* of incidents (deaths, injuries, and device malfunctions) through the years and *across six major surgical specialties* of gynecology, urology, general, colorectal, cardiothoracic, and head and neck surgery. We characterized the *potential causes* for incidents and measure their *impact on patients* and *on the progress of surgery*.

There have been previous studies on safety and effectiveness of robotic surgery based on the experience of different surgical institutions as well as analyses of the FDA MAUDE data. However, an important question that is unanswered is whether the evolution of the robotic systems

with new technologies and safety features *over the years* has improved the *safety of robotic systems* and their *effectiveness across different surgical specialties*.

Most of the previous work only focused on two common surgical specialties of gynecology and urology, analyzed small number of failures experienced by specific surgical teams or small subsets of the MAUDE data, or studied specific types of device failures, e.g., electro-cautery failures, injuries caused by electrosurgical units, or individual instrument failures (see Section 3.5.5 for more details).

We used MedSafe to automatically *retrieve all the reported events* on robotic surgical systems from the MAUDE database and *extract important safety information* from the reports, including patient complications, surgical specialties and types of robotic procedures, most common types of system malfunctions, and the actions taken by the surgical teams to recover from failures. We found that: (i) the overall numbers of injury and death events per procedure have stayed relatively constant over the years, (ii) the probability of events in complex surgical specialties of cardiothoracic and head and neck surgery has been higher than other specialties, and (iii) device and instrument malfunctions have affected thousands of patients and surgical teams by causing complications and prolonged procedure times.

MedSafe enables large-scale analysis of nationwide incidents reported to the FDA over any timing period and, thus, facilitates more accurate estimation of the prevalence of incidents over the years and more effective evaluation of robotic surgical systems across different surgical specialties. Our goal is to use the knowledge gained from the analysis of past incidents to provide insights on design of future robotic surgical systems that by taking advantage of advanced safety mechanisms, improved human machine interfaces, and enhanced safety training and operational practices can minimize the adverse impact on both the patients and surgical teams.

3.5.1. Methods

We extracted the reports related to the systems and instruments used in minimally invasive robotic surgery by searching for related keywords (e.g., device and manufacturer names) in the *Device Name* and *Manufacturer Name* fields of over 2.9 million MAUDE records posted between January 2000 and December 2013. That led us to an initial list of adverse event reports, from which we filtered out those with duplicate database keys (reporting the same adverse event for multiple devices). In addition to the structured information that was directly available from the reports, we

extracted further information from the unstructured human-written descriptions of events by natural language parsing of the *Event Description* and *Manufacturer Narrative* fields. As shown in Figure 3.5, the MedSafe adverse event analysis tool combines domain knowledge with linguistic rules to interpret the semantics of the event descriptions. This is done by creating several domain-specific dictionaries (e.g., for patient complications, surgery types [75], surgical instruments [76], and malfunction types) as well as syntactic rules, parts-of-speech (POS) taggers, and negation detectors [31]. The results generated by each step of our automated analysis were manually reviewed for accuracy and validity.

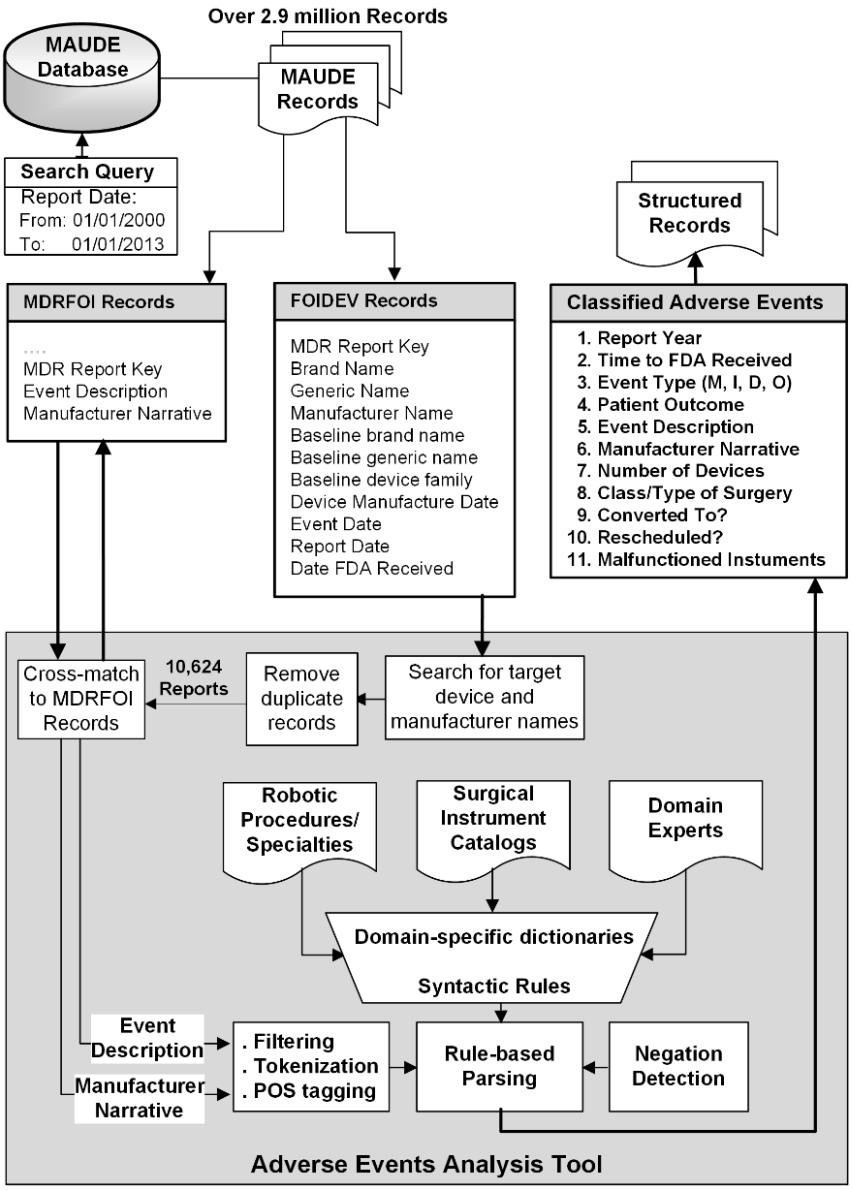


Figure 3.5. Data extraction and analysis flow from the FDA MAUDE database.

More specifically, we extracted the following information from the unstructured text in the reports:

- **Patient injury** (such as burns, cuts, or damage to organs) and death events that were reported under another *Event Type*, such as "Malfunction" or "Other".
- **Surgical specialty and type of robotic procedure** during which the adverse events occurred.
- **Major types of device or instrument malfunctions** (e.g., falling of burnt/broken pieces of instruments into patients' bodies or electrical arcing of instruments).
- **Adverse events that caused an interruption in the progress of surgery**, by leading the surgical team to troubleshoot technical problems (e.g., restarting the system), convert the procedure to non-robotic surgical approaches (e.g., laparoscopy or open surgery), or abort the procedure and reschedule it to a later time.

The dictionaries of keywords for surgery types/specialties and surgical instruments were constructed based on the online information available on the da Vinci surgeries [75] and instruments catalog [76] from the manufacturer.

We compared the number of adverse events (in general) and injury/death events and procedure conversions (in particular) per 100,000 procedures across different surgical specialties. The rate of events was estimated by dividing the number of adverse events that occurred in each year (based on the *Event Date*) by the annual number of robotic procedures performed in the United States. The total number of procedures per year was extracted from the device manufacturer's reports [72], [77] for 2004–2013, as shown in Figure 3.6. For 2010–2013, the annual numbers of procedures performed in the United States were extracted directly from the annual reports of the manufacturer [72]. For 2004–2009, we estimated the numbers of procedures by measuring the graphs in the company's investor presentations [77]. Whenever the estimated numbers from two different sources did not match or the data was available only for the total worldwide procedures, we chose the maximum number of procedures for that year in order to achieve a lower bound on the likelihood of events. We further estimated the number of procedures per week from annual number of procedures by fitting a 4-degree polynomial curve (R2 = 0.999) to the bar graph of annual procedures and calculating the area under the fitted curve for every week (see Figure 3.6).

The annual number of procedures per surgical specialty was available only for gynecology, urology, and general surgery after 2007. So we estimated a combined annual number of procedures

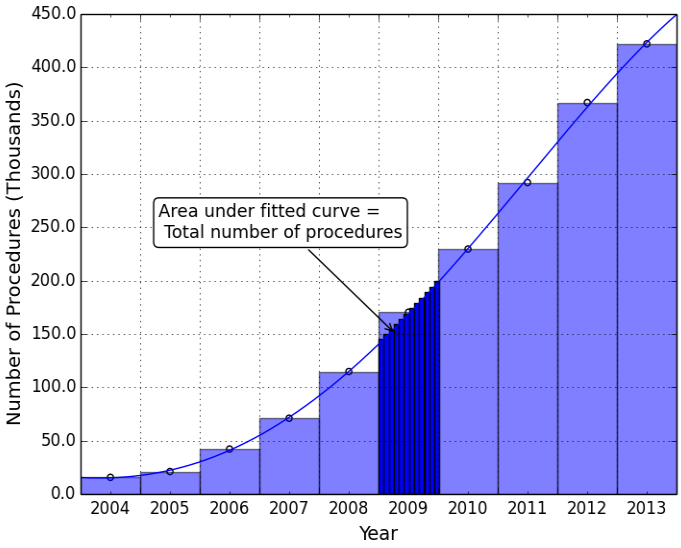


Figure 3.6. Estimated numbers of procedures performed during 2004–2013.

for cardiothoracic and head and neck surgery by assuming that the majority of the remaining procedures (other than genecology, urology, and general) were related to these specialties, as, according to the manufacturer reports, they are the only other specialties for which the robot has been used [72].

We assumed that the rate of underreporting for injury and death events are low and are independent from the type of surgery, because the device manufacturers are required and monitored by the FDA to report serious injury and death events to the MAUDE database. However, due to possible changes in the reporting rates during the years, the total number of events per procedure in the whole study period was compared across different surgical specialties. The two-sided P values (< 0.05) and 95% confidence intervals were used to determine the statistical significance of the results. The cumulative number of malfunctions per procedure was used to evaluate the trends in malfunction rates over 2004–2013.

To characterize the major causes to which injury and death events were attributed by the reporters, we performed a manual review of event descriptions for all the reports made before 2013.

3.5.2. Results

We identified a total of 10,624 events related to the robotic systems and instruments, reported over 2000–2013. About 98% of the events were reported by device manufacturers and distributors,

and the rest (2%) were voluntary reports. In the same period, over 1,745,000 robotic procedures were performed in the United States, so the estimated number of adverse events per procedure was less than 0.6% (95% confidence interval (CI), 0.6–0.62).

Data included 1,535 (14.4%) adverse events with significant negative patient impacts, including injuries (1,391 cases) and deaths (144 cases), and over 8,061 (75.9%) device malfunctions. For the rest of the events (1,028 cases), the *Event Type* information either was not available or was indicated as "Other." We identified 160 adverse events (1.5%) that included some kind of patient injuries but were reported as a "Malfunction" or "Other."

3.5.2.1. Trends in Adverse Event Reports

Figure 3.7 shows the overall trends in the annual numbers of reports and the estimated rates of events per 100,000 procedures over 2004–2013. The left Y-axis corresponds to the bars showing the absolute numbers of adverse events (based on the years that reports were received by the FDA). The right Y-axis corresponds to the trend lines showing (in logarithmic scale) the annual number of adverse events per 100,000 procedures (based on the year the events occurred). Numbers on the bars indicate number of deaths reported per year. Error bars represent 95% confidence intervals for the proportion estimates. Because of the small number of injury and death events reported for 2004 and 2005, a combined rate was calculated for 2004–2006. Of all the reported events, 40 were related to the published articles or the legal disputes received by the manufacturer.

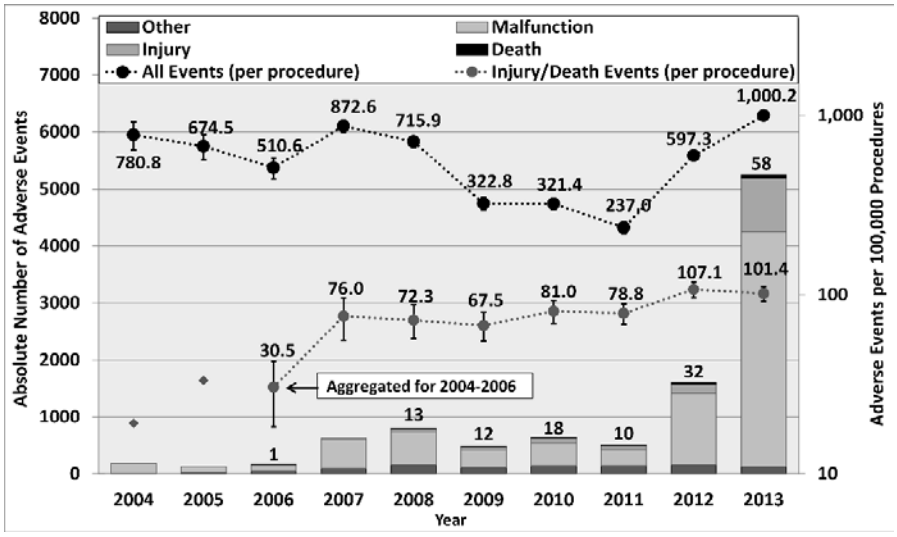


Figure 3.7. Annual numbers of adverse event reports and rates of events per procedure.

We made the following observations based on these results:

- The absolute number of reports made per year has significantly increased (about 32 times) since 2006, reaching 58 deaths, 938 patient injuries, and 4,124 malfunctions in 2013. The number of robotic procedures performed per year has increased 10-fold in the same period [77].
- While the annual average number of adverse events was about 550 per 100,000 procedures (95% confidence interval (CI), 410–700) between 2004 and 2011, in 2013 it peaked at about 1,000 events per 100,000 procedures (one event reported in every 100 procedures).
- The numbers of injury and death events per procedure have stayed relatively constant since 2007 (mean = 83.4 per 100,000 procedures, 95% CI, 74.2–92.7).

3.5.2.2. Adverse Events across Different Surgical Specialties

Table 3.6 shows the numbers of adverse events reported in different surgical specialties and their impact on patients (injuries or deaths) and progress of surgery (procedure conversion or rescheduling). The last row shows examples of the most common types of procedures reported in each specialty.

- The majority of reports were related to gynecology (30.1%), urology (14.7%), and cardiothoracic (3.7%) surgeries, such as hysterectomy (2,331), prostatectomy (1,291), and thoracic (110) procedures, respectively. The higher percentage of adverse events in gynecologic and urologic surgeries could be explained by the higher number of these procedures performed (86% of all the robotic procedures performed in the United States) compared to other surgical specialties (less than 14.2% of all procedures) [77].
- Cardiothoracic and head and neck surgeries involved a higher number of deaths per adverse event report (6.4% and 19.7%) than gynecology and urology (1.4 and 1.9%).
- The highest number of procedure conversions per adverse event was for cardiothoracic (16.8%) and urology (13.5%), and the highest rates of procedure rescheduling were for urology (9.5%), general (3.0%), and cardiothoracic (2.8%) surgeries.

Of all the reports, only 5,721 (53.8%) indicated the class and type of surgery involved. However, the majority of reports with missing information on the type of surgery were related to device malfunctions and “Other” events (97.6%). In order to compare the rate of adverse events across different specialties, we focused only on reports related to injuries, deaths, and procedure

conversions. For the majority of these events (92.2% of injury reports, 95.1% of deaths, and 72.2% of procedure conversions), the surgery type information was available and the rest (with "N/A" surgical specialty) were removed from our analysis. In order to estimate the rate of events per procedure, we regrouped the events into four major categories of "Gynecology," "Urology,"

Table 3.6. Adverse events in different surgical specialties:
Deaths, injuries, malfunctions, procedure conversion or rescheduling, common types of surgery.

	No. (%) [95% Confidence Interval]						
	Gynecology	**Urology**	**Cardio-thoracic**	**Head & Neck**	**Colorectal**	**General**	**N/A**
Overall [a]	3,194 (30.1) [29.2–31.0]	1,565 (14.7) [14.0–15.4]	393 (3.7) [3.3–4.1]	71 (0.7) [0.5–0.9]	301 (2.8) [2.5–3.1]	197 (1.9) [1.6–2.2]	4,903 (46.2) [45.3–47.1]
Event Type [b]							
Death	46 (1.4) [1.0–1.8]	30 (1.9) [1.2–2.6]	25 (6.4) [4.0–8.8]	14 (19.7) [10.4–29.0]	11 (3.7) [1.6–5.8]	11 (5.6) [2.4–8.8]	7 (0.1) [0.0–0.2]
Injury	818 (25.6) [24.1–27.1]	272 (17.4) [15.5–19.3]	64 (16.3) [12.6–20.0]	14 (19.7) [10.4–29.0]	58 (19.3) [14.8–23.8]	56 (28.4) [22.1–34.7]	109 (2.2) [1.8–2.6]
Malfunction	2,103 (65.8) [64.2–67.4]	902 (57.6) [55.2–60.0]	226 (57.5) [52.6–62.4]	35 (49.3) [37.7–60.9]	209 (69.4) [64.2–74.6]	110 (57.8) [48.9–62.7]	4,476 (91.3) [90.5–92.1]
Other	227 (7.1) [6.2–8.0]	361 (23.1) [21.0–25.2]	78 (19.8) [15.9–23.8]	8 (11.3) [3.9–18.7]	23 (7.6) [4.6–10.6]	20 (10.2) [6.0–14.4]	311 (6.3) [5.6–7.0]
Conversion	236 (7.4) [6.5–8.3]	212 (13.5) [11.8–15.2]	66 (16.8) [13.1–20.5]	6 (8.5) [2.0–15.0]	29 (9.6) [6.3–12.9]	14 (7.1) [3.5–10.7]	217 (4.4) [3.8–5.0]
Rescheduling	26 (0.8) [0.5–1.1]	148 (9.5) [8.1–10.9]	11 (2.8) [1.2–4.4]	1 (1.4) [0–4.1]	1 (0.3) [0–1.0]	6 (3.0) [0.6–5.4]	77 (1.6) [1.3–1.9]
Common Surgery Types	Hysterectomy (2,331)	Prostatectomy (1,291)	Thoracic (110)	Thyroidectomy (19)	Cholecyst-ectomy (118)	Hernia repair (37)	
	Myomectomy (328)	Nephrectomy (138)	Lobectomy (67)	Tongue base resection (19)	Colectomy (61)	Nissen fundoplication (34)	
	Sacrocolpopexy (170)	Pyeloplasty (31)	Mitral valve repair (54)	Transoral robotic (18)	Low anterior resection(44)	Gastric bypass (28)	
	Oophorectomy (120)	Cystectomy (48)	Coronary artery bypass (23)		Colon resection(25)	Gastrectomy (15)	

[a] Percentages are over all the adverse event reports (n = 10,624).

[b] Percentages are over the total adverse events reported for a surgical specialty.

"General," and "Cardiothoracic and Head and Neck," according to the manufacturer's reports [77]. The "General" category includes both colorectal and general specialties.

As shown in Table 3.7, for cardiothoracic and head and neck surgery, the rates of injuries, deaths, and procedure conversions have been significantly higher than other specialties. During 2007–2013, the estimated rate of deaths have been 52.2 per 100,000 procedures for cardiothoracic and head and neck specialties vs. 5.7 in gynecology, urology, and general surgeries (RR = 9.23, 95% CI, 6.35–13.40, P < 0.0001). Also, the rate of injuries and procedure conversions in these specialties have been 91.0 and 89.7 per 100,000 procedures vs. 71.5 (RR = 1.27, 95% CI, 0.99–1.63, P < 0.052) and 29.2 (RR = 3.07, 95% CI, 2.38–3.97, P < 0.0001) in the other surgical categories.

Table 3.7. Comparsion of adverse events rates in different surgical specialities (2007 - 2013).

	No. (rate per 100,000 procedures) [a] **[95% CI]**		**Cardiothoracic and Head and Neck vs. Gynecology, Urology, and General**	
	Gynecology, Urology, General	**Cardiothoracic, Head and Neck, Other**		
Total Procedures	1,661,891	74,709	**Relative Risk (95% CI)**[b]	***P* Value**
Total Adverse Events	5,209	447		
Event Type				
Death	94 (5.7)	39 (52.2)	9.23 (6.35–13.40)	< 0.0001
Injury	1188 (71.5)	68 (91.0)	1.27 (0.99–1.63)	< 0.052
Conversion	485 (29.2)	67 (89.7)	3.07 (2.38–3.97)	< 0.0001
Rescheduling	180 (10.8)	12 (16.1)	1.48 (0.83–2.66)	< 0.19 [c]

a Percentages are over total number of procedures in each column.

b Assuming that the level of underreporting across different surgical specialties is similar.

c Not statistically significant because of the small number of samples (12) in the cardiothoracic and head and neck surgery.

3.5.2.3. Device and Instrument Malfunctions

We identified five major categories of device and instrument malfunctions that impacted the patients, either by causing injuries and complications or by interrupting the progress of surgery and/or prolonging procedure times. Table 3.8 shows the numbers of events in each category, the event types as indicated by reporters (including Malfunction (M), Injury (IN), Death (D), and Other (O)), and the actions taken by the surgical team to resolve the problems. The malfunction categories and actions taken by the surgical teams are not mutually exclusive, and in many cases two or three different malfunctions or two actions were reported in a single event. Figure 3.8 uses

Venn diagrams to depict the intersections among different malfunction categories and actions taken by the surgical team.

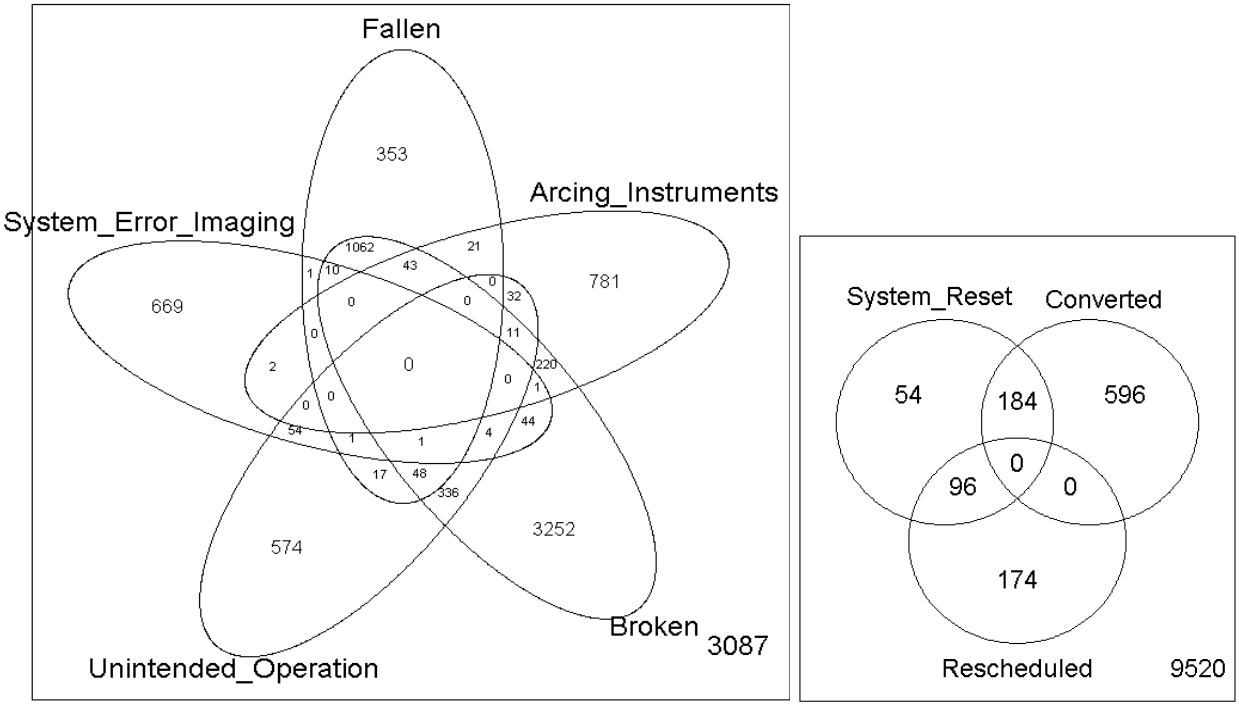
Table 3.8. Major categories of malfunctions and surgical team actions.

Malfunction Category	Description	No. of Reports (% of all)					Surgical Team Actions (% of malfunction category)		
		Total	Event Type [a]				System Reset	Procedure Converted	Procedure Rescheduled
			M	IN	D	O			
System Errors	- System error codes/faults - System transferred into a recoverable or non-recoverable safety state	536 (5.0%)	44	23	1	468	231 (43.1%)	330 (61.6%)	133 (24.8%)
Video/ Imaging Problems	- Loss of video feed - Display of blurry images at surgeon's console or assistant's touchscreen	275 (2.6%)	21	18	0	236	53 (19.3%)	145 (52.7%)	94 (34.2%)
Broken Pieces Falling Into Patients	- Burnt or broken parts, instruments, components - Fell into surgical field or patient body cavity - Additional procedure time was spent to find/remove pieces from patient body	1,557 (14.7%)	1,396	119	1	41	3 (0.2%)	38 (2.4%)	5 (0.3%)
Broken Tip Covers/ Elec. Arcing	- Tears, burns, holes on tip covers (insulation failure) - Electrical arcing, sparking, charring of instruments	1,111 (10.5%)	900	193	0	18	2 (0.2%)	18 (1.6%)	0 (0.0%)
Unintended Instrument Operation	- Unintended or unstoppable movements without surgeon's command - Instruments not working, opening/closing - Instruments not recognized by system	1,078 (10.1%)	919	52	2	105	31 (2.9%)	93 (8.6%)	21 (1.9%)
Other	- Cable, wire, tube, or instrument damages and breakages - Issues with electrosurgical units, power supplies/cords, patient-side manipulators - Other "Malfunction" events	5,092 (47.9%)	4,962	55	1	74	20 (0.4%)	62 (1.2%)	13 (0.3%)
Total (% of all)	- All malfunctions	9,377 (88.3%)	8,061	443	5	868	305 (3.3%)	630 (6.7%)	246 (2.6%)
	All Adverse Events	10,624 (100%)	8,061	1,391	144	1,028	334 (3.1%)	780 (7.3%)	270 (2.5%)

- **System errors and video/imaging problems** contributed to 787 (7.4%) of the adverse events and were the major contributors to procedure interruptions, including system resets

(274 cases, 82% of all system resets), conversion of the procedures to a non-robotic approach (462 cases, 59.2% of all conversions), and aborting/rescheduling of the procedures (221 cases, 81.8% of all cases). System errors are raised upon detection of device problems that cannot be autonomously recovered from and either require manual system reset (recoverable errors) or stopping the procedure (non-recoverable errors). Table 3.9 lists the descriptions and frequencies of the most common error codes extracted from the reports.

- **Falling of the broken/burnt pieces into the patient's body** constituted about 1,557 (14.7%) of the adverse events. In almost all these cases, the procedure was interrupted, and the surgical team spent some time searching for the missing pieces and retrieving them from the patient (in 119 cases, a patient injury, and in one case a death, was reported).
- **Electrical arcing, sparking, or charring of instruments** and burns or holes developed in the tip cover accessories constituted 1,111 reports (10.5% of the events), leading to nearly 193 injuries, such as burning of tissues.
- **Unintended operation of instruments**, such as uncontrolled movements and spontaneous powering on/off, happened in 1,078 of the adverse events (10.1%), including 52 injuries and two deaths.



(a) (b)

Figure 3.8. Intersections among different (a) Malfunction categories and (b) Surgical team actions.

A total of 3,067 adverse event reports were not classified by MedSafe in any of the malfunction categories. For 9,520 adverse events, no system resets, conversions, or rescheduling were reported.

The *Other* category represents the malfunctions that could not be classified in any of the classes, including cable and instrument breakages that did not lead to other types of malfunctions, electrosurgical unit and power supply problems, and patient-side manipulator issues.

In total, 9,377 reports were about technical problems, including 1,104 cases (10.4% of all the adverse events) in which the procedure was interrupted and additional time was spent on troubleshooting the errors, resetting the system, and/or converting the procedure to a traditional technique, or rescheduling the procedure to a later time. In 1,019 of cases (10.9% of all the malfunctions), the device or instrument malfunction was detected prior to start of the procedure, of which in 20 cases the procedure was rescheduled to a later time and in two cases it was converted to a non-robotic approach.

Figure 3.9 shows the cumulative rates of malfunctions per procedure over 2004–2013. Overall, the malfunction rates decreased after 2006, but the rate of cases with arcing instruments and broken instruments followed a relatively constant trend. The sudden increase in the rate of broken instruments after the middle of 2012 could be related to changes made to the adverse event reporting practices by the manufacturer in 2012 (mostly related to instrument cable breaks) [78], as well as increased reporting of adverse events after concerns about the safety of robotic surgery were raised by the FDA [79], [80] and public media in early 2013 [81] – [83].

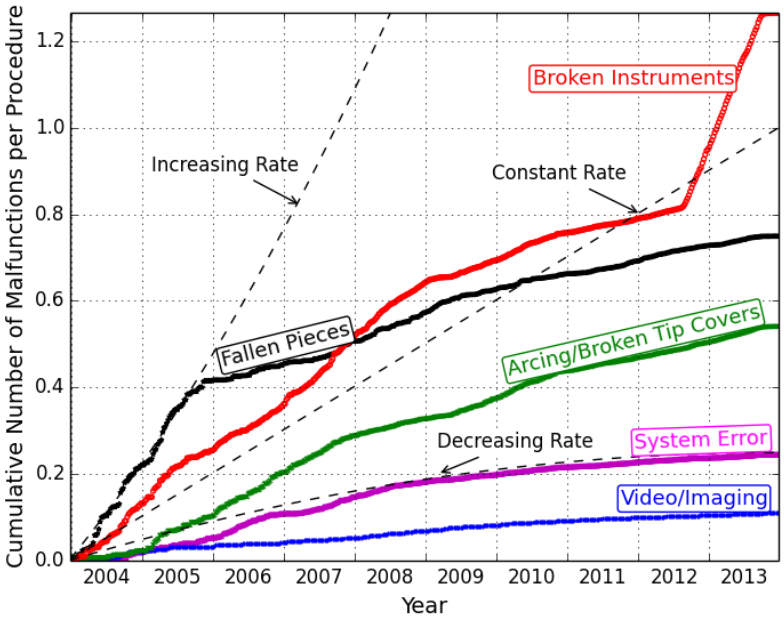


Figure 3.9. Cumulative rates of malfunctions per procedure.

The rates of malfunctions per procedure were obtained for each week (see Figure 3.6 for more details on the estimation of the number of procedures).

Table 3.9. Most frequent system error codes raised by the surgical system.

System Error Code	Description	Type of Safe State that System Transits To	No. of Adverse Events
#20008	The angular position of one or more robotic joint's on the specified manipulator, as measured by the joint's primary control sensor (encoder) and the secondary sensor (potentiometer), were out of specified tolerance for agreement.	Recoverable	62
#23008			42
#20013			34
#23013			20
#21008			18
#21013			17
#23002			8
#20009			7
#22003			5
#212	A voltage-tracking fault reported by the digital signal processor (dsp) when the actual voltage to drive current through the motors deviates from the expected voltage by a specified amount.	Non-recoverable	31
#23	Hardware wheel "wdog" has tripped on one of the digital communication links in the system (due to an excessive number of retries on hardware message packets). This means that the system cannot reliably communicate over that digital link and therefore cannot continue normal operation. Communication faults in the low-voltage differential signal carrying information about the patient side manipulator. Communication faults between two system components.	N/A	28
#1	A power supply voltage was out of range.	Non-recoverable	19
#3	A redundant switch was missing its ground sense, or the contacts did not report as expected at startup.	N/A	15
#23017	A motor did not respond as expected, and the measured motion did not match the internal stimulation of the motor.	Recoverable	14
#2	A reference voltage was out of range.	N/A	14
#31030	One of the camera controller units in the doco has failed to power on after multiple attempts or has shut down after initially powering up.	N/A	14
#5	One or more fans are not moving as desired	N/A	10
#297	An electronic component was reporting an incorrect configuration.	Non-recoverable	9
#252	Master supervisory controller did not receive an expected message within a specified time.	Recoverable	8
#23020	One of the switches in a specific manipulator is showing inconsistent signals on its two switch leads.	Recoverable	7
#25589	During the power up self-test, the remote arm controller board (rac) brakes failed the brake voltage test.	Recoverable	6
#25588	A sympathetic error and occurs during the self-test upon system power-up when a loop response test fails.	Recoverable	6
#23007	On startup, one or more robotic joints on the manipulator did not make the prescribed test motion to within the specified tolerance.	Recoverable	6
#21003	The arm did not perform the commanded motions during startup within a specified tolerance.	N/A	5
#281	A processor did not complete a step during system startup within the allotted time.	Non-recoverable	5
#23034	After a specified amount of time, a valid event was not seen for one of the remote compute engine switches.	Recoverable	5
#45049	A communication timeout with the software running the da Vinci onsite application.	Recoverable	4

3.5.2.4. Injury and Death Causes

A manual review of a sample set of injury and death reports (from 2000–2012) allowed us to classify the causes indicated by reporters into three main categories: inherent risks associated with surgery, technical issues with the robot, and mistakes made by the surgical team. For the majority of death events, little or no information was provided in the reports. About 50% of the death events were indicated by the reporters to be related to inherent risks or complications of surgery, 11.6% due to patient's history or health state, and 7% were attributed to surgeon/staff mistakes (e.g., incorrect instrument change or accidental cuts of artery). Of all the reported deaths in different classes of surgery, at least 75.3% (64 of 86) happened after the procedure (mainly due to patient history, infection/sepsis, or uncontrollable and heavy bleeding) and 17.4% (15 of 86) happened during the procedure. Of the deaths that occurred during the procedures, five were due to inadvertent cuts or punctures of organs and the others were related to complications such as uncontrolled bleeding, pulmonary embolism, and cardiac arrest.

As Table 3.10 shows, about 62% of the injury events involved device malfunctions and the rest involved operator errors (7.1%), improper positioning of patient or port incisions (6.3%), inherent risks of surgery (3.9%), or problems with grounding the equipment (1.5%).

Table 3.10. Summary of death and injury reports (2000–2012).

	Death Reports (Total = 86)	
	Example Causes	**Num. of Reports (%)**
	Surgeon/staff mistake	6 (7.0%)
	Patient's history	10 (11.6%)
	Inherent risks and complications	43 (50.0%)
	N/A	27 (31.4%)
During Procedure	Punctures, bleeding, pulmonary embolism, cardiac arrest	64 (75.3%)
After Procedure	Infection/sepsis, heavy bleeding	15 (17.4%)
Injury Reports (Total = 410)		
Example Causes		**Num. of Reports (%)**
Device malfunctions[a]		254 (62.0%)
Surgeon/staff mistake		29 (7.1%)
Improper positioning of patient led to post-operation complications such as nerve damage		17 (4.1%)
Inherent risks of surgery and patient history		16 (3.9%)
Burning of tissues near port incisions		9 (2.2%)
Possible passing of electrosurgical unit currents through instruments to the patient body		6 (1.5%)
Surgeon felt shocking at the surgeon-side console		2 (0.5%)
N/A		77 (18.8%)

The following are the most common flawed operational practices used by the surgical team that contributed to catastrophic events during surgery:

- Inadequate experience with handling emergency situations
- Lack of training with specific system features
- Inadequate troubleshooting of technical problems
- Inadequate system/instrument checks before procedure
- Incorrect port placements
- Incorrect electro-cautery settings
- Incorrect cable connections
- Inadequate manipulation of robot master controls
- Inadequate coordination between hand & foot movements
- Incorrect manipulation or exchange of instruments

A more comprehensive analysis of multidimensional causes of incidents is the topic of the future research.

3.5.3. Reasons for Recalls and Recovery Actions

We also extracted 19 recalls of the da Vinci surgical system and instruments reported to the FDA from January 2000 to December 2012. While only a small number of recalls were issued by the company over the years, they impacted a large number of devices (109,709 devices and instruments) on the market. After the concerns raised by the FDA and the public about increase in the number of adverse event reports, the manufacturer issued 13 new recalls in the 8-month period of April-November 2013 alone. Table 3.11 and Table 3.12 list examples recalls of the da Vinci (S) Control System and Instruments; we further classified the recalls, based on their reasons, into four categories of *software*, *electrical*, *computer control*, and *mechanical* problems.

Important insights into the safety issues of the robot can be extracted from the recalls, because the data includes the actual technical problems confirmed by the manufacturer that may present potential harm to patients, as well as the recovery actions taken to address the failures.

The following are our findings:

- Of all the recalls, ten were reported for the robot's control system, affecting about 3,741 systems on the market, while the rest (nine out of 19) were related to accessories and instruments used with the robot, affecting about 105,968 devices.
- The majority (seven out of 10) of control system recalls (Table 3.11) were due to computer- and electrical-related malfunctions (affecting 1568 devices), but the recalls due to mechanical malfunctions (three out of 10) affected a larger number of devices (2173) on the market.
- The problems with the robot control system were often handled at a very high cost to both the manufacturer and the hospitals.
- Software and mechanical issues were addressed by sending field system engineers to all the locations to update or repair the systems (in about 1,500 devices).
- Hospitals were advised to have backup equipment and instrumentation available and to be prepared to convert to alternative surgical techniques (mentioned in [84] and in the system's user manual, according to [85]), costing about $2 million per back-up device and instruments.

The manufacturer's recommendations that healthcare providers continue to use the device until the corrective system updates or service visits are made (e.g., for recall numbers Z-2204-2008 and Z-2930-2011 in Table 3.11) and that they use the backup systems in the case of failures, do not reflect advisable practices. Many of the reported failures might be repeatable and in the time until the service visit, there is potential for system downtime, prolonged procedures, or patient injuries in all devices that are affected by the same defect (259 devices in these two examples). Additionally, some of the problems such as a software defect that can lock up the system cannot be resolved even by having redundant backup devices. It is well-understood in software engineering that two versions of the same software may well experience the same technical problems [86].

Table 3.11. Recalls of da Vinci (S) endoscopic instrument control system (2000–2012).

Recall Record Numbers	Device Name (Model Number)	Date	Malfunction Type	Reason for Recall	Company Recovery Action	Num. of Devices
Z-1244-2007	da Vinci S A4.3 SW (Model IS1200)	Sep 26 2007	Computer Software	Under certain conditions, the product's software may crash and require a manual override or restart before functioning again.	Service Visit + Software Upgrade	405
Z-0079-2008	da Vinci S (Model IS2000)	Feb 21 2008		System lock-up: Software anomalies could cause product failure during use; or on start-up. System transitions to a safe “soft-lock” state.	Software Upgrade	159
Z-2204-2008	da Vinci S (Model IS2000)	Sep 16 2008		Defective software chip may cause the system to fail and lock up.	Urgent Letter + Replace Chip	112
Z-1245-2007	da Vinci S (Model IS2000) (Auxiliary power board (APB))	Feb 22 2008	Electrical	Product may malfunction and fail to start up on AC power.	Service Visit	38
Z-0151-2008 Z-0152-2008	da Vinci S (Model IS2000) (Vision Cart Model VS2000)	Feb 22 2008		Under-rated fuses may be installed which will result in fuse failure and a loss of power to the vision cart and any ancillary equipment connected to it.	Notification Letter + Correction	63
Z-1180-2008	da Vinci S (Model IS2000) (Revision A51_P5)	June 12 2008	Computer Control	Delay in responding: In certain circumstances, the device may not respond immediately to a user’s command, such as master clutch or camera control.	Notification Letter + Correction	9
Z-1161-2010	da Vinci S (Model IS2000) (Revision A51_P7)	Apr 05 2010		Gripper or scissor jaws may close inadvertently, and will not open on command, and various other reported modes of failure. Control by surgeon may fail, and this failure may be difficult to detect.	Notification Letter + Service Engineer Visit	782
Z-0670-2007	da Vinci S 4 Arm (Model IS2000)	Mar 29 2007	Mechanical	Spinal pin could limit mechanical motion of the arm and make system unavailable for surgery.	Rework as Part of Routine Service Visit	24
Z-2930-2011	da Vinci Si (Model IS3000)	Aug 03 2011		Potential failure of the retention component of the Master Tool Manipulator (MTM) could cause uncontrolled movement.	Urgent Correction Letter + Component Retrofit	183
Z-1202-2012	da Vinci S, da Vinci Si, da Vinci Si-e	Mar 13 2012		The holding brake may allow passive uncontrolled motion due to gravity during specific power-off conditions.	Urgent Correction Letter + Instructions	1966

Table 3.12. Example recalls of da Vinci accessories (2000–2012).

Recall Record Numbers	Device Name (Model Number)	Date	Malfunction Type	Reason for Recall	Company Recovery Action	Num. of Devices
Z-0660-2007	8mm EndoWrist Bipolar Maryland Instrument	Mar 28 2007	Computer Software	The product was incorrectly programmed as training instruments, allowing it to be used for 30 surgical procedures instead of 10.	Urgent Letter + Return	8
Z-1811-2008	da Vinci S Cardiac Probe Gasper Instrument (For model IS2000)	Sep 17 2008		There is a software interface problem that will not allow the IS2000 system to recognize the instrument, which causes the loss of operability of the instrument; delay in surgery; and loss of dexterity.	Urgent Letter + Return	11
Z-0723-05	da Vinci 8 mm EndoWrist Curved Scissors	Apr 22 2005	Mechanical	Blades on the scissor may break and separate from the main unit as a result of corrosion damage.	Notification Letter	278
Z-0669-2008	da Vinci S 5 mm Instrument Cannula	Jan 31 2008		5mm Cannula may have sharp edges on the inner diameter that may cause particulate shavings to be skive (scraping) from the instrument shafts during surgery.	Urgent Notification Letter + Instructions	89
Z-0657-2008 Z-0658-2008 Z-0659-2008		Jan 31 2008		There may be a ridge on the side of the cannula which has the potential to abrade instrument shafts and generate black particulate matter.	Urgent Notification Letter + Replace	896
Z-1348-2008	da Vinci S 8mm Instrument Cannula (model IS2000)	Aug 06 2008		(1) Incorrect dimension on Luer on smoke evacuation cannulae not allowing for secure attachment function. (2) Incorrect labeling.	Urgent Letter + Remove	39
Z-2104-2012 Z-2103-2012 Z-2101-2012 Z-2102-2012 Z-2105-2012 Z-2106-2012	da Vinci S 4 Arm Disposable Accessory Kit (model IS3000)	Jul 27 2012		Specific lots of the Instrument Arm Drapes were manufactured with a sterile adaptor that may have difficulty engaging an instrument.	Urgent Recall Letter + Return	92,390
Z-0258-2008	da Vinci 8 mm EndoWrist PK Dissecting Forceps	Jan 24 2008	Labeling and Sterilization	Mislabeling-electrical isolation requirements: devices were incorrectly labeled with a CF symbol (suitable for direct cardiac application), not their proper BF Symbol on the instrument housing.	Urgent Letter + Instructions + Update Labeling	1,136
Z-2339-2012	Tip Cover for 8m Monopolar Curved Scissors (Disposable)	Sep 10 2012		There is potential for the sterility of the product to be compromised.	Notification Letter + Instructions	11,121

Notwithstanding the fact that existing da Vinci robotic systems already have built-in safety and recovery mechanisms, the reported failures are identifiable single points of failures that could be prevented or recovered from at much lower costs and in a more timely fashion. The software lockups can be resolved by using a technique called rollback recovery (or *check-pointing*), a standard technique that have been shown to be effective in tolerating 70–90% of hardware and software faults [87], [88]. Other examples include the use of redundant components in the system design, real-time error detection and reconfiguration strategies for automatic replacement of defective system components, and timely software updates.

3.5.4. Limitations

The results of our study come with the caveats that inherent risks exist in all surgical procedures (more so in complex procedures) and that the MAUDE database suffers from underreporting and inconsistencies. Thus, the estimated number of adverse events per procedure are likely to be lower than the actual numbers in robotic surgery. The sensitivity of adverse event trends to changes in reporting mechanisms, surgical team expertise, and inherent risks of surgery could not be assessed based on this data.

Further, the lack of detailed information in the reports makes it difficult to determine the exact causes and circumstances underlying the events. The example causality analysis results presented here are limited to our knowledge of the system and the information provided in the MAUDE reports. There might be other system hazards and causal factors involved in the adverse events that could be determined using STAMP, but were missed in our analysis.

3.5.5. Related Work

In this section we provide a summary of the related work on safety and effectiveness of minimally invasive robotic surgery and its comparison to non-robotic minimally invasive surgical methods (laparoscopy), as well as an overview on the FDA MAUDE database and previous analyses of adverse events in robotic surgery.

3.5.5.1. Robotic versus Non-Robotic Surgical Methods

Previous studies on the effectiveness of robotic surgery and comparison of the outcomes to non-robotic minimally invasive methods fall into one of the following categories: (1) case-controlled

studies that performed retrospective comparison of outcomes from robotic and non-robotic minimally invasive procedures done by specific surgical teams and institutes, and (2) meta-analysis studies that systematically reviewed and combined the results from multiple studies. There are only a few small-sized randomized controlled trial studies that compared robotic and laparoscopic minimally invasive methods (e.g., one study compared robot-assisted laparoscopic radical prostatectomy (RALP) and retropubic radical prostatectomy (RRP) [89]). The case-controlled studies may suffer from selection bias in choosing the surgery methods for individual patients. Usually the less difficult cases are chosen early in the learning curve of a new procedure method, potentially leading to an unfair comparison of procedures. In addition, retrospective studies based on analysis of medical records may underreport actual rates of complications due to missing information in the records. Further, the unfavorable outcomes experienced during learning curve may not get published at all, leading to biased analysis of outcomes by meta-analysis studies [90].

Table 3.13 provides a summary of the previous studies across different surgical specialties [90] – [106]. For each specialty, we selected a sample of studies on the most common type of procedure performed in that specialty (e.g., hysterectomy in gynecology, prostatectomy in urology, and mitral valve repair in cardiology). We picked the case-controlled and meta-analysis studies that covered a large population of patients (n > 100 when possible) as well as randomized control trials, from the high impact journals published between 2007 and 2014.

As Table 3.13 shows, most of the studies especially in gynecology and urology, for which the robots are extensively used, show better outcomes compared to other minimally invasive methods in terms of amount of blood loss during surgery, length of hospital stay, and mortality rates.

However, the case-controlled studies report contradictory results on the mean operative time and complication rates in robotic versus laparoscopic hysterectomy and prostatectomy. This is because those factors are highly dependent on the expertise level of surgeon and the number of cases required to overcome the learning curve in robotic surgery [107]. Further, comparisons of outcomes for more complex procedures in cardiothoracic and head and neck surgery have rarely been done, and the existing studies often show that robotic approach is no more effective than non-robotic minimally invasive methods.

Table 3.13. Related work on comparison of robotic vs. non-robotic minimally invasive surgical methods.

Specialty	Study	Study Type	Surgical Procedure	Patients (n)	Mortality Rate (%)	Complication Rate (%)	Conversion/ Reoperation Rate (%)	Mean Operative Time(min)
Gynecology	Boggess et al. 2008 [91]	Cohort	Hysterectomy LAP	81	--	13.6	4.9	213.4
			ROB	103	--	5.8	2.9	191.2
	Gaia et al. 2010 [90]	Systematic Review	LAP	424	--	3.8	9.9	209
			ROB	396	--	2.0	4.9	219
	Wright et al. 2012 [92]	Database Analysis	LAP	2,464	0.2	9.8	--	--
			ROB	1,437	0.1	8.1	--	--
	Wright et al. 2013 [93]		LAP	4,971	0	5.3	0.1	--
			ROB	4,971	0	5.5	0.1	--
Urology	Rozet et al. 2007 [94]	Multiple Surgeons	Prostatectomy LRP	133	0	9.1	0	160
			RRP	133	0	19.4	3.0	166
	Berryhill et al. 2008 [95]	Meta Analysis	LRP	5,411	0	15.6	1.5	227
			RRP	5,472	1 death	6.6	0.5	164
	Porpiglia et al. 2013 [96]	Randomized Control Trial	LRP	60	--	11.6	--	138.1
			RRP	60	--	16.6	--	147.6
	Robertson et al. 2013 [97]	Systematic Review	LRP	4,952	0	0.76	0.3	238
			(*Predicted Prob.) RRP	6,768	0.2*	0.06	0.9*	225
General	Rawlings et al. 2007 [98]	Single Surgeon	Colectomy LAP	27	--	14.8	7.4	199.4
			ROB	30	--	20.0	6.6	225.2
	Müller-Stich et al. 2007 [99]	Randomized Controlled Trial	Fundoplication LAP	20	--	10.0	0	102
			ROB	20	--	15.0	0	88
	Breitenstein et al. 2008 [100]	Prospective Case-matched	Cholecystectomy LAP	50	--	2.0	0	50
			ROB	50	--	2.0	0	55
	Edelson et al. 2011 [101]	Retrospective Database Analysis	Gastric banding LAP	120	--	16.6	2.5	30.9
			ROB	287	--	17.1	3.1	91.5
Cardiothoracic	Mihaljevic et al. 2011 [102]	Single Institute	Mitral valve repair PST	270	0	9.9	2.6	277
			ANT	114	0	2.7	2.6	327
			ROB	261	0	11-12	9.1	387
	Swanson et al. 2014 [103]	National Database Analysis	Lobectomy VATS	295	--	18.98	--	253.8
			(*Major events) ROB	295	--	16.95*	--	269.4
			Wedge resection VATS	325	--	15.69	--	171.6
			(*Major events) ROB	325	--	21.58*	--	195.6
	Kent et al. 2014 [104]	National Database Analysis (Propensity-matched)	Lobectomy VATS	1,233	1.1	45.3	--	--
			ROB	411	0.2	43.8	--	--
Head & Neck	Lee et al. 2011 [105]	Retrospective Single Institute	Thyroidectomy END	96	--	10.4	0	142.7
			ROB	163	--	11.0	0	110.1
	Yoo et al. 2012 [106]	Retrospective Single Institute	END	165	--	11.5	0	145.2
			ROB	45	--	12.9	0	118.3

ROB: robotic, LAP: laparoscopy, LAPT: laparotomy LRP: laparoscopic radial prostatectomy, RRP: robot-assisted radial prostatectomy, PST: Partial sternotomy, ANT: Mini-anterolateral thoracotomy, VATS: Video-assisted thoracic surgery

3.5.5.2. Failures of Robotic Surgical Systems

There have been several reports by individual surgical institutions on the various software-related, mechanical, and electrical failures experienced before or during robotic procedures [108] – [121]. Table 3.14 summarizes these reports by providing the number and types of procedures performed at each center as well as the failure rates and number of cases in which failures led to conversion or rescheduling of procedures. The rates of device malfunctions and failure-related conversions reported by these studies varied between 0.4-8.0% and 0.1-2.7%, with an average of 3% (95% confidence interval (CI), 1.9–4.2) and 0.9% (95% confidence interval (CI), 0.4–1.4), respectively.

3.5.5.3. Surgical Adverse Events Reports from the MAUDE Database

Table 3.15 shows summary of related work ([122] – [128]) on analysis of the FDA adverse event reports on robotic surgical systems. All these studies were performed by *manual* extraction and review of *subsets of the MAUDE data*. Almost half of the studies only focused on *specific types of device failures* (e.g., electro-cautery failures [122], electrosurgical injuries [123], and instrument failures [124]) and only considered the *gynecology and urology* specialties in their analysis. None of the previous studies considered the total volume of procedures performed in their analysis period and none of them assessed the risk of adverse events across different surgical specialties and their impact on the progress of surgery.

Table 3.14. Summary of related work on failures of robotic surgical systems.

Study (Year)	Surgery Types	Medical Institute	No. Cases	Total Number of Failures (Failure Rate) Types of Malfunctions	Converted	Rescheduled
Eichel [108] (2005)	Urologic	UC Irvine	200	Total = 5 (2.5%) Software (4), Mechanical (1)	Laparoscopic (1) (0.5%)	N/A
Kozlowski [109] (2006)	Radical Prostatectomy (RLRP)	Virginia Mason Medical Center (VMMC)	130	Total = 6 (4.6%) Setup joint (2), Software incompatible (1), Robotic arm malfunction (1), Power-off error (1), Monitor loss (1)	Laparoscopic (1) Open (1) (1.5%)	4 (3.1%)
Borden [110] (2007)	Laparoscopic Prostatectomy	Virginia Mason Medical Center (VMMC)	350	Total = 9 (2.6%) Setup joint (2), Robotic arm (2), Camera (1), Power error (1), Console metal break (1), Software incompatible (1), Monitor loss (1)	Laparoscopic (1) Open (2) (0.9%)	6 (1.7%)
Zorn [111] (2007)	Radical Prostatectomy (RLRP)	University of Chicago, School of Medicine (2003–2006)	725	Total = 7 (0.96%) (Recover. = 0.21%, Non-Recover. = .05%) Power-up failure (1), Optical malfunction (3), Robotic arm (1), Camera (2)	Surgeon handicap (3)	4 (0.5%)
Fischer[112] (2008)	Radical Prostatectomy	Klinik Hirslanden, Switzerland	210	Total = 2 (1%) Robotic arm (2)	Laparoscopic (2) (1.0%)	N/A
Lavery [113] (2008)	Radical Laparoscopic Prostatectomy (RALP)	11 Institutions 700 Surgeons	8,240	Total = 34 (0.4%) Robotic arm (14), Optical system (14), Master malfunctions (4), Power supply/circuit (6), Unknown error (3)	Laparoscopic (2) Open (8) (0.1%)	24 (0.3%)
Ham [114] (2009)	Radical Laparoscopic Prostatectomy	Yonsei University, Korea	1	Case report of Surgeon's console failure	Delayed 15 min	
Kim [115] (2009)	Urology, General, Obstetrics and Gynecology, Thoracic and Cardiac Otorhinolaryngology	Yonsei University College of Medicine, Korea (2005–2008)	1,797	Total = 43 (2.4%) **Robot failures (24):** On/off failure (1), Console malfunction (5), Robotic arm (6), Optic system (2), System error (10) **Instrument failures (19):** Shaft injuries (9), Wire cutting (2), Unnatural motion (2), Instrument tip (2), Limitation in motion (1)	Laparoscopic (2) Open (1) (0.2%)	N/A
Kaushik [116] (2010)	Robot-assisted Radical Prostatectomy (RARP)	Survey of 176 Surgeons from 4 Countries	N/A	Total failures = 260 (before or after surgery) Robotic arm (38%), Camera (17.6%), Setup joint (13.8%), Power error (8.8%), Ocular monitor loss (8%), Instruments (7.6%), Console handpiece break (3%), Software (1.9%), Backup battery (0.3%), Instrument identification (0.3%)	Open (18.8%), Laparoscopic (15%), Another robot, with one fewer robotic arm (8.7%)	46 (57.5%)
Finan [117] (2010)	Gynecologic Oncology	University of South Alabama (2006–2008)	137	Total = 11 (8%) Robotic arm (2), Light or camera cord (2), Maylard bipolar (1), Power failure (1), Port problem (1), Others (3)	Delayed 25 min.	N/A
Mues [118] (2011)	Urology, Gynecology, Cardiothoracic, General surgery, Otolaryngology, Neurosurgery	Ohio State University Medical Center (2008–2009)	454	Tip cover failures = 12 (2.6%) Significant patient complications (25%)	Repaired at the time of surgery	N/A
Agcaoglu [119] (2012)	General Surgery	Cleveland Clinic	223	Total = 10 (4.5%) Robotic instrument (4), Optical system (3), Robotic arms (2), Robotic console (1)	Open surgery (6) (2.7%)	N/A
Chen [120] (2012)	Urological Surgery	Veterans General Hospital, Taiwan (2005–2011)	400	Total = 14 (3.5%) Robotic arm/joint (11), Optical system (1), Power system/connector (1), Endoscopic instrument (1), Software incomp. (1)	Recoverable(10) Laparoscopy (3) (0.8%)	1
Buchs [121] (2014)	General Surgery	A Teaching Institution (2006–2012)	526	Total = 18 (3.4%) Robotic instruments (9), Robotic arms (4), Surgical console (3), Optical system (2)	Laparoscopic (1) (0.2%)	N/A

Table 3.15. Related work on analysis of the FDA adverse event reports on robotic surgical systems.

Study	No. Reports (Years)	System under Study	Surgical Specialties	Major Results
Murphy et al. [122]	38 system failures 78 adverse events (2006 – 2007)	da Vinci system	N/A	Most of these events were related to broken instrument tips or failures of electrocautery elements.
Andonian et al. [125]	189 (2000 – 2007)	ZEUS and da Vinci systems	N/A	Estimated failure rate of 0.38% for robotic-assisted laparoscopic surgeries.
Lucas et al. [126]	1,914 (2003 – 2009)	da Vinci system models dV and dVs	N/A	Both device malfunctions and open conversions were reduced by increased robotic experience and newer surgical systems. The number of patient injuries did not change and the number of deaths increased.
Fuller et al. [123]	605 (2001 – 2011)	da Vinci system	N/A	24 (3.9%) of reports were related to electrosurgical injuries (ESI), of which 37.5% resulted in surgical intervention.
Friedman et al. [124]	565 (2009 – 2010)	da Vinci Instruments	N/A	The majority of events were related to the instrument wrist or tip (285), 174 were related to cautery problems, 76 were shaft failures, and the rest were cable and control housing failures (36).
Gupta et al. [127]	741 (2009 – 2010)	da Vinci system	Urology Gynecology	The events were related to the use of energy instruments (43.5%), surgical systems (19.3%), and the instruments (11.7%). The severity of events was correlated with the type of surgery and the type of device.
Manoucheri et al. [128]	26 injuries 24 deaths (2006 – 2012)	da Vinci system	Gynecology	The majority of injuries (65%) were not directly related to use of robot; 21% were related to operator error; and 14% were due to technical system failures.

3.5.6. Discussion

Our analysis shows an increasing number of adverse events related to the robotic surgical systems being reported. As cautioned by the FDA [64], [79], the number of MAUDE reports may not be used to evaluate the changes in rates of events over time, because the increased reporting of events may be due to different factors, e.g., the increasing use of surgical systems [77], changes in the manufacturers' reporting practices [78], and/or better awareness and increased publicity resulting from product recalls, media coverage, and litigation [79]. Therefore, we measured the prevalence of adverse events in each year by estimating the number of events reported per procedure. We found that despite a relatively high number of reports, the vast majority of

procedures were successful and did not involve any problems and the number of injuries/death events per procedure has stayed relatively constant since 2007. However, total number of malfunctions reported per procedure (0.46%, 95% confidence interval (CI), 0.45-0.47%) was about six times lower than the average number of malfunctions per procedure (3%, 95% confidence interval (CI), 1.9–4.2) published by different surgical institutions (see Table 3.14). Also the total number of injuries and deaths reported per procedure (0.08%, 95% confidence interval (CI), 0.08-0.09%) was about the same as the predicted complication rates for robotic surgery [97], but an order of magnitude less than the lowest rate of complications reported for robotic surgery in previous studies (2% [100]) (see Table 3.13). This further confirms the uncertainty in the rates of events due to possible underreporting and changes in reporting practices. However, the non-negligible percentage of system-related incidents that negatively impacted patients (in terms of injuries and deaths (14.4%) and procedure interruptions (10.4%)) suggests that a better appreciation and understanding of the nature of adverse events is required.

Our analysis shows that estimated number of events per procedure in complex surgical areas, such as cardiothoracic and head and neck surgery were significantly higher than gynecology, urology, and general surgeries. Although not all the reported injuries and deaths were due to device problems, and the procedure conversions, of themselves, cannot be considered adverse events [129], [130], the estimated numbers of injury/death events and conversions per procedure can be used as a metric to measure the difficulty experienced in different surgical specialties. The best that we can tell from the available data is that the higher number of injury, death, and conversion *per adverse event*, in cardiothoracic and head and neck surgeries, could be indirectly explained by the higher complexity of the procedures, less frequent use of robotic devices, and less robotic expertise in these fields. Although the use of robotic technology has rapidly grown in urology and gynecology for prostatectomy and hysterectomy, it has been slow to percolate into more complex areas of cardiothoracic and head and neck surgery. Between 2007 and 2013, over 1.4 million (86%) robotic procedures in gynecology and urology were performed in the United States, while the number of procedures in other surgical specialties altogether was less than 250,000 (14.2%) [72], [77]. The limitations of the robotic user interface [131], long procedure times [132], steep learning curve [133], [134], and higher costs for purchase and maintenance of robotic systems and instruments [135] are some factors that may have contributed to the lower utilization of the robotic approach in more complex surgical procedures. For example, only a select

type of robotic cardiac procedures are reported to have been successfully performed using the robots, such as mitral valve repair and internal mammary artery harvest [136] – [138]. The recent experiences of highly competent robotic teams that performed multi-vessel coronary artery bypass grafting (CABG) showed that the robotic approach may be associated with higher mortality and morbidity rates compared to open surgery [139].

In practice, the use of the robotic platform involves the interface of a sophisticated machine with surgical teams and patients, in an area of patient care that is safety-critical. A recent FDA survey on a sample of experienced robotic surgeons highlighted some of the biggest challenges in robotic surgery as: hand-eye coordination, use of foot pedals at the console, the system setup procedures, learning of platfrom (e.g., port and arm insertion), and training of surgical staff [140].

There are similar safety-critical considerations in other industries, such as the commercial aircraft, in which the interfaces of complex machines and human operators are vital to the well-being of travelers. While a direct comparison between those two is neither possible nor advisable, we can emulate the manner in which the aviation industry, government, academia, and society at large have come together in creating standards and procedures to achieve a continuously high-level of safety and mission-time reliability of better than 10^{-9} for electronic equipment in commercial aircrafts. Some of the factors that have been critical to that success include:

- Careful analysis of accidents (by airline authorities and the National Transportation Safety Board (NTSB)) to ensure that mistakes are not repeated and designs are continually improved.
- Extensive use of hazard analysis and sophisticated safety design techniques and controls.
- Oversight and certification by government authorities, e.g., Federal Aviation Administration (FAA).

Many safety-related aspects of robotic surgery, such as device and operators' certification, accident investigation, and safety hazards, could be compared to aviation industry (see Table 3.16). From a technology perspective, employing substantially improved safety practices and controls in the design, operation, and validation of robotic surgical systems could prevent some of the reported events. Some examples include:

- New safety engines for monitoring of procedures (including surgeon, patient, and device status) and providing comprehensive feedback to surgical team on upcoming events and troubleshooting procedures to prevent long procedure interruptions.

Table 3.16. Comparison between two safety-critical industries: aviation vs. robotic surgery.

	Aviation	Robotic Surgery
Operation: Type Device Targets	 Semi-autonomous Airplanes Passengers	 Semi-autonomous Robots Patients
Age	80 years (approx. 1934)	< 20 years (approx. 1999)
Certification: Administrated by: -Device -Operator -Others	 Federal Aviation Administration (FAA) -Aircraft certified under 14 CFR 121 -Pilots certified by privilege levels -Crew certified by airlines	 Food and Drug Administration (FDA) -Robot approved by 510K -Surgeons trained but not certified -Staff trained but not certified
Training	Required by FAA for pilots	Provided by company for surgeons
Accidents	All accidents investigated by NTSB and other authorities based on the evidence collected from the site of accident	Reported by the users and company to the FDA MAUDE database, on a voluntary basis
Safety Hazards	-Natural: Weather conditions, fire, etc. -Mechanical/Electrical: Engine, electromagnetic interference, etc. -Humans: Incorrect info by control center, pilot/crew errors, passenger misuses, hijacks	-Natural: Patient history/condition/procedure -Mechanical/Electrical: Arm malfunctions, system errors, etc. -Humans: Incorrect info by the company for setup/troubleshooting, pilot/staff mistake,

- Providing real-time feedback to the surgeon on the safe surgical paths that can be taken or safety barriers that prevent the robotic tools to enter to certain portions in the surgical workspace [141], based on the patient-specific anatomical models, as well as surgeon-specific modeling and monitoring of robotic surgical motions [142], to minimize the risk of approaching dangerous limits and inadvertent patient injuries.
- Improved human-machine interfaces and surgical simulators that train surgical teams for handling technical problems [50], [143] and assess their actions in real-time during the surgery.
- Improved mechanisms for logging and reporting of incidents experienced during procedures to enable more accurate validation of safety and effectiveness of surgical systems.

3.6. Conclusions

We applied the STAMP modeling framework to system-theoretic causality analysis of adverse events in safety-critical medical devices, exemplified by robotic systems in minimally invasive surgery. Specifically, we used STAMP to identify examples of unsafe interactions among components and human operators at different layers of the system control structure and determined

potential flaws in the design and operation of system that led to unsafe system states leading to incidents. To exemplify the value of STAMP, we selected three relevant reports (marked with "*" in Table 3.3) on accidents that occurred during gynecologic, cardiothoracic, and urologic procedures. In these examples, a burning or damage of tissue due to applying incorrect amount of electro-cautery occurred. Each of the reports specified that the bipolar cautery was incorrectly connected to the monopolar connection of the electrosurgical unit. Our experience is that frequently root cause analysis of such an accident ends up in blaming the surgical assistants for not performing the robotic setup correctly. STAMP provided us with a detailed view on all the potential causal factors involved and identified two unsafe actions that were performed by the surgeon and surgical assistant in these three events: (i) the surgical assistant connected the bipolar cautery to a monopolar connection on the electro-cautery unit and (ii) the surgeon cauterized the tissue using the incorrect electro-cautery. Some of the potential causal and contextual factors that led to those actions include: (i) the robot was not designed to provide enough feedback on the status of connections and the surgeon had an inaccurate mental model about the connections, (ii) the electro-cautery unit was not designed by adequate safety mechanisms to prevent such mistakes and the surgical assistant might not have been well-experienced or distracted at the time. This analysis shows that design of both the robot and electro-cautery unit as well as the communication protocols and training for the surgeons and surgical assistants should be improved in order to prevent similar events in the future.

Our experience with STAMP shows that the systems-theoretic accident analysis techniques do not only enable an *in-depth understanding of adverse event causes*, but can also *improve the reporting of adverse events by guiding how to investigate and report* the important factors involved in the accidents. Furthermore, the systems-theoretic causality models can be *used by the manufacturers and regulatory organizations to identify the possible hazards and causal factors leading to accidents* early in the design and approval process to mitigate adverse patient impacts by design of proper safety monitors and control mechanisms.

To enable system-theoretic causal analysis of larger sets of adverse event reports, we proposed a new ontology model based on human-in-the-loop control structures that guides automated extraction of safety-related information from the textual description of events. We used this ontology model for automated analysis of adverse events in MedSafe.

Our analysis of adverse events in robotic surgery using MedSafe, demonstrated several important findings. While the robotic surgical systems have been successfully adopted in many different specialties, (i) the overall numbers of injury and death events per procedure have stayed relatively constant over the years, (ii) the probability of events in complex surgical specialties of cardiothoracic and head and neck surgery has been higher than other specialties, and (iii) device and instrument malfunctions have affected thousands of patients and surgical teams by causing complications and prolonged procedure times.

As the surgical systems continue to evolve with the new technology, uniform standards for device approval, surgical team certification and training, advanced human machine interfaces, improved accident investigation and reporting mechanisms, and safety-based design techniques should be developed to reduce incident rates in the future.

CHAPTER 4
SAFETY HAZARD SIMULATION FOR RESILIENCY ASSESSMENT AND SAFETY TRAINING

4.1. Overview

In our analysis of over 10,000 adverse events related to robotic surgical systems, we found that 9,377 (88.3%) of the reported events involved device and instrument malfunctions (see Section 3.5.2.3). Those events had significant negative patient impacts, occasionally leading to deaths and injuries or causing procedure interruptions to troubleshoot system problems. Although state-of-the-art robotic surgical systems are designed with safety mechanisms that try to detect those failures and put the system into a recoverable or non-recoverable safe state, in practice those mechanisms are imperfect. In particular, out of 536 system errors detected during procedures, 488 (91%) led to interruption in progress of surgery, forcing the surgical teams to manually restart the system (43% of 488), convert the procedure (61.5%), or reschedule it to a later date (24.8%). In some cases after several system resets, the procedure was converted or rescheduled. This is mainly due to two factors:

(i) The diagnostic mechanisms are not comprehensive enough to correctly identify the causes of malfunctions and system errors during surgery. Thus, information on the type of system error (e.g., in an error condition of recoverable or non-recoverable) and corresponding troubleshooting procedures might be inadequate or are incorrectly communicated to the surgical team. The root causes are often determined only after the fact, when further investigations by the field service engineers are performed and the failure scenarios are replicated. For example, an encoder or sensor malfunction was reported as a recoverable system error during a procedure, when it was not recoverable and could only be fixed by replacing the component after the procedure (e.g., MAUDE report 3035720 [145]).

This chapter contains material from the published works [143], [144], coauthored with D. Chen, X. Li, A. Lewis, Z. Kalbarczyk, R. K. Iyer, J. Raman, T. Kesavadas, and N. Leveson, copyrighted by ACM and Springer.

(ii) System operators (including surgeons, surgical assistants, and field service engineers) are often not well trained to correctly interpret reasons for observed system errors and to choose efficient troubleshooting actions to recover from emergency situations. For example, in one event, it was reported that the surgical team spent a significant amount of time troubleshooting a non-recoverable system error while the patient was under anesthesia for more than one hour (see report No. 1743065 [146]).

These results show the importance of designing robust safety features and verifying the effectiveness of detection and recovery mechanisms to assist surgical teams in predicting and preventing critical events and performing effective troubleshooting procedures. Also, training of system operators should be improved to make troubleshooting and handling of adverse events a central part of the training experience. In fact, in other safety-critical industries (e.g., commercial aviation) great effort is spent on continually improving safety practices by careful investigation of accidents, extensive hazard analysis, and advanced safety design, combined with comprehensive simulation-based training that includes operation in the presence of safety-critical failures [147].

The international safety standards (e.g., ISO 14971 for medical devices [20] and ISO 26262 for automobiles [148]) recommend identifying potential safety hazards and defining safety requirements to implement mechanisms that can detect and mitigate hazards. The standards (e.g., NASA Software Safety Guidebook [149]) also emphasize the importance of fault-injection testing as a means to validate the robustness of safety mechanisms in the presence of faults and abnormal conditions [150].

As discussed in Chapters 1 and 2, traditional hazard analysis techniques used for medical devices primarily focus on the failures of individual components or human errors in the system. Other potential causal factors, such as complex software errors and unsafe component interactions, are often not thoroughly considered during the analysis. Systems-theoretic hazard analysis techniques such as STPA (Systems-Theoretic Process Analysis) [12] overcome this limitation by modeling accidents as complex dynamic processes resulting from inadequate control mechanisms that violate safety constraints. It is shown that STPA can identify additional causes for accidents that are not detected by FTA and FMEA techniques [12], [56].

Software-implemented fault injection (SWIFI) [151], [152] is a common technique for validating the effectiveness of fault-tolerance mechanisms by studying the behavior of the system in the presence of faults. The effects of software or hardware faults are emulated by randomly

changing code or data at different software locations. However, with the increasing size of software in today's complex systems, it is a challenging task to define specific fault types and locations that can effectively emulate realistic fault scenarios.

In this chapter, we present a systems-theoretic approach to empirically validate the robustness of safety mechanisms in medical cyber-physical systems by identifying the critical locations within the system to target software fault injection. More specifically, we use the potential causes of safety violations identified by STPA to define types and locations of faults to be injected in software in order to assess the resiliency of the system under realistic hazard scenarios. This is achieved by developing a safety hazard injection framework that can be integrated with the robot control software in:

(i) an actual system, to emulate the control flaws identified using STPA hazard analysis and conduct resiliency assessment and validation of the safety mechanisms during the design and implementation phases.

(ii) a simulated environment, to assess the impact of safety hazards without causing adverse impacts on the electrical and mechanical components of the actual system.

In the second scenario, representative safety hazard scenarios extracted from past incident data can be simulated in a virtual environment for simulation-based safety training of system operators.

As a case study, we use the RAVEN II telerobotic surgical system [14], [15] and develop a safety hazard simulation platform composed of a surgical simulator integrated with a software-based fault-injection engine, which automatically inserts faults into different modules of the robot control software. We evaluate the feasibility of the proposed framework using examples of real adverse events from the FDA MAUDE database. The source code for the surgical simulator based on the RAVEN II robot and the hazard injection engine is publicly available at [153].

4.2. Background

4.2.1. RAVEN II Robotic Surgical Platform

Surgical robots are designed as human-operated robotically controlled systems, consisting of a *teleoperation console*, a *robot control system*, and a *patient-side cart* (which hosts the robotic arms, holding the surgical endoscope and instruments). The most critical component of the robot control is the electronic control system, which is responsible for the following:

- *Receiving the surgeon's commands* issued using master manipulators and foot pedals on the teleoperation console.
- *Translating the surgeon's commands* into the corresponding surgical robot movements.
- *Providing video feedback* of the surgical field (inside patient's body) to the surgeon through 3D vision on the teleoperation console.
- *Performing safety checks* on to ensure the safe operation of the surgical robot.

The RAVEN II robot is an open-source platform for research in teleoperative robotic surgery [14], [15]. Figure 4.1 depicts a typical configuration of a robotic telesurgery system, composed of a master console, communication channel, and a RAVEN II surgical robot, including software and hardware components. The master console provides the means for the surgeon to issue commands to the robot using foot pedals and master tool manipulators. The desired position and orientation of robotic arms, foot pedal status, and robot control mode are sent from the master console to the robot control software over the network using the Interoperable Teleoperation Protocol (ITP), a protocol based on the UDP packet protocol [154]. The control software receives the user packets, translates them into motor commands, and sends them to the control hardware, which enables the movement of robotic arms and surgical instruments. The robot consists of two cable-driven surgical manipulators attached with tool interfaces and the instruments. Each surgical manipulator is operated by DC motors and has seven degrees of freedom.

Figure 4.2(a) shows the main hardware and software modules in the RAVEN II control system. The control software runs as a node (process) on the Robotic Operating System (ROS) middleware [157] on top of a real-time (RT-Preempt) Linux kernel. There are three main threads

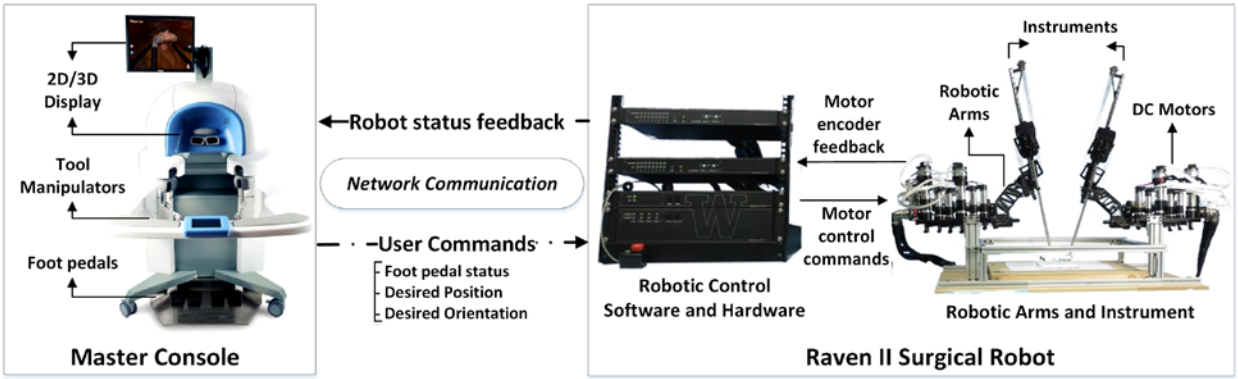
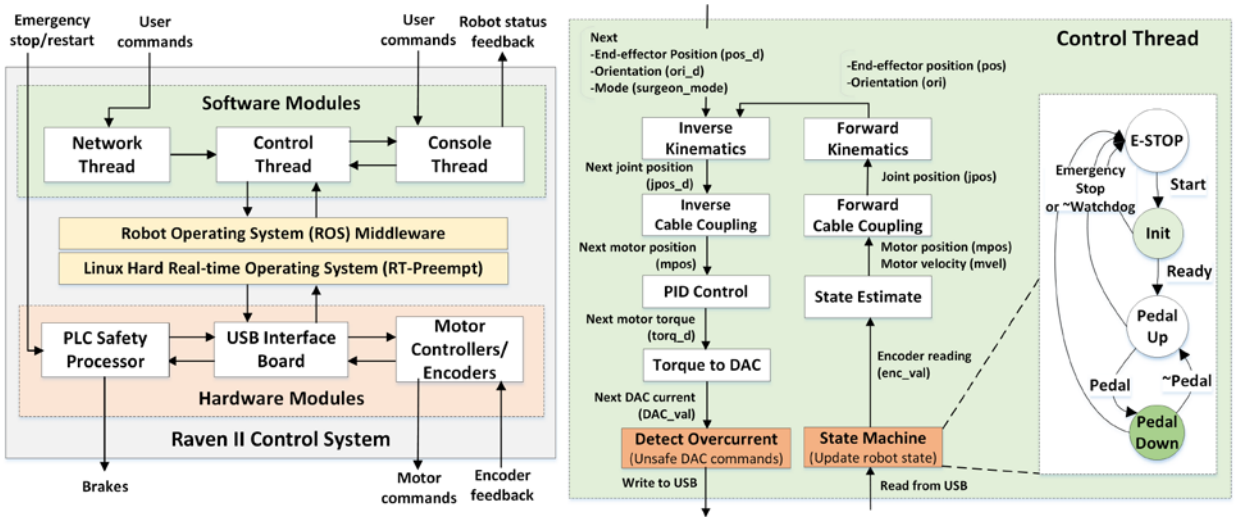


Figure 4.1. Robotic telesurgery using RAVEN II surgical platform (modified from [155], [156]).



(a) (b)

Figure 4.2. RAVEN II control system: (a) Software and hardware modules [15], [156], (b) Computation steps in the control thread (kinematics chain of RAVEN control) and the safety state machine [156]

running in parallel in the RAVEN control software: (i) the network layer thread which receives the command packets from the master controller over network; (ii) the control thread where the robot's kinematics and control computations are performed; and (iii) the console thread which provides an interface for setting the control modes and displaying robot's status to user.

The control software communicates with the motor controllers and a Programmable Logic Controller (PLC) through two custom 8-channel USB interface boards. The interface boards include commodity programmable devices, digital to analog converters, and encoder readers. The motor controllers send movement commands (torque values calculated based on the desired joint positions) to the DC motors and read back the encoder values from the motors (to estimate the current joint positions). The PLC controls the fail-safe brakes on the robotic joints and monitors the system state by communicating with the robotic software.

Figure 4.2(b) shows the computation steps in the software control thread performing the kinematic chain of the RAVEN control. The operator commands are sent to the control software as incremental motions (desired end-effector positions (*pos_d*) and orientations (*ori_d*)). The current end-effector's configurations (*pos* and *ori*) are calculated using forward kinematics function. The inverse kinematics calculates the joint (*jpos_d*) and motor (*mpos_d*) positions that are required to obtain the desired end-effector configurations and positions. Finally, the amount of torque needed for each motor to reach its new position is obtained from a Proportional-Interal-

Derivitive (PID) controller. The PID controllers are commonly used in control systems to minimize the error between the measured variables (here the current motor positions and velocities) and the desired ones (next motor positions and velocities) by adjusting the control commands (here the motor torque commands). The motor torques are transferred in the form of DAC commands (*DAC_value*) from the control software to the motor controllers on the USB boards, to be executed on the motors [14][156].

Both the control software and the PLC operate in a state machine that consists of four states: (a) emergency stop ("E-STOP"), (b) initialization ("Init"), (c) foot pedal released ("Pedal Up"), and (d) foot pedal pressed ("Pedal Down"), as shown in Figure 4.2(b). The control software's state is synced with the PLC state every one millisecond through the USB interface boards. At power-up, both software and PLC are at the Emergency-Stop ("E-STOP") state, the motor brakes are engaged, and motor controllers are stopped. The system goes through the initialization phase before getting ready for the operation. During the initialization phase, the mechanical and electronic components of the system are tested to detect any faults or problems and each robotic arm moves from its resting position into the surgical field. After successful initialization, the robot enters the "Pedal Up" state, in which the robot is ready for teleoperation but the brakes are engaged and robot does not move. When the foot pedal is pressed by the operator, the robot moves to the "Pedal Down" state, where the brakes are released, allowing the teleoperation console to control the robot [14].

The RAVEN II robot has the following safety mechanisms:

- A physical start button should be pressed to take the robot out of the "E-STOP" state. At any time pressing the emergency stop button will immediately stop the robot by putting the PLC and control software into "E-STOP" state.
- Whenever the human operator lifts their foot from the pedal, the system enters the "Pedal Up" state and engages the fail-safe power-off brakes on the motors and disengages the master console from manipulating the surgical arms.
- The control software performs safety checks on the motor controller commands before they are sent to the USB I/O boards. These safety checks (Detect Overcurrent block in Figure 4.2(b)) compare the electrical current commands sent to the digital to analog converters (DACs) with a set of pre-defined thresholds to ensure the motors and arm joints do not move beyond their safety limits.

- The control software sends a periodic ("I'm alive") square-wave watchdog signal to the PLC through the USB boards. Upon detecting any unsafe motor commands, the control software stops sending the watchdog signal. The PLC safety processor monitors the periodic watchdog signal from software and in absence of the watchdog signal puts the system into the "E-STOP" state.

4.2.2. Systems-Theoretic Hazard Analysis Using STPA

STPA is a hazard analysis technique based on the STAMP accident causality model and driven by concepts in systems and control theories. As discussed in Chapter 3, STAMP is an accident modeling and causality analysis framework that treats safety as a control problem rather than a component failure problem. In STAMP, accidents are modeled as dynamic processes resulted from inadequate enforcement of safety constraints on the behavior of components at different layers of the system control structure [12].

STPA hazard analysis starts by identifying the potential accidents and system-level hazards associated with those accidents, and the safety requirements (constraints) that must be controlled

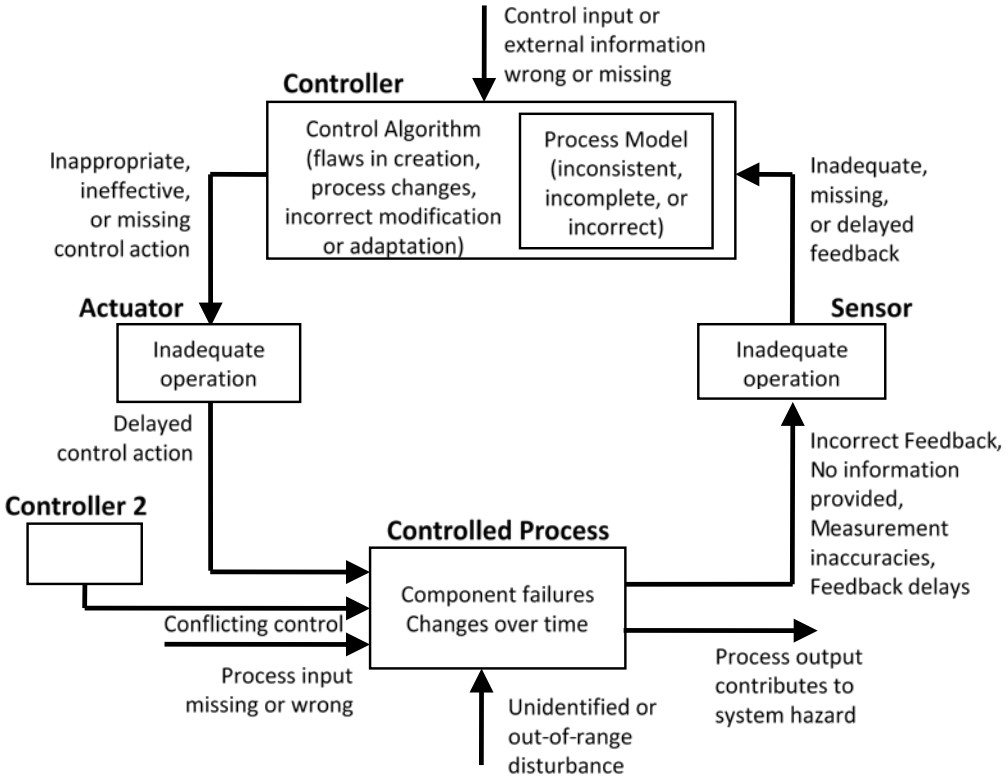


Figure 4.3. Potential causes for unsafe control in a generic control loop (as defined and depicted in [12]).

for the system. Then the unsafe control actions for each component in the control structure that can lead to system hazards are identified. There following scenarios are considered for identifying the unsafe control actions: (i) a required control action is not performed, (ii) a control action is performed in a wrong state, leading to a hazard, (iii) a control action is performed at an incorrect time (too late, too early, or in the wrong order), (iv) a control action is performed for an incorrect duration (too long or too short), and (v) a control action was provided, but not followed by the controlled process. Finally, the potential causes for those hazardous control actions are determined by examining the operation of components and their interactions in each loop of the control structure. Figure 4.3 shows the common types of control flaws in a generic control loop that can be used for identifying the potential casual factors.

4.3. Systems-Theoretic Safety Validation Using Fault Injection

We employed systems-theoretic hazard analysis using STPA to identify the safety hazards of a typical robotic telesurgical system and the potential causal factors that might lead to safety violation in the RAVEN II system. Then we validated the robustness of the safety mechanisms of the RAVEN II robot to the safety hazard scenarios identified using STPA by simulating their causal factors using software-implemented fault injection.

4.3.1. Safety Hazards and Unsafe Control Actions

Figure 4.4(a) shows the high-level control structure of robotic surgical systems (also shown in Figure 3.2). Software and hardware control loops (outlined by dashed lines) are further refined in Figure 4.4(b) to illustrate the functional details of components (control algorithms, process models) and interactions (control actions and feedback) among them in the software and hardware controllers of the RAVEN II surgical robot. Every controller uses an algorithm to generate the control actions based on the current process model. The control actions taken by each controller (e.g., motor commands and status sent by the software controller) changes the state of the controlled process (e.g., the motor controllers) and the process model of the other controller (e.g., the PLC). The feedback (e.g., motor encoder values) sent back from the controlled process and the other controller update the process model used (e.g., current joint status) by the controller.

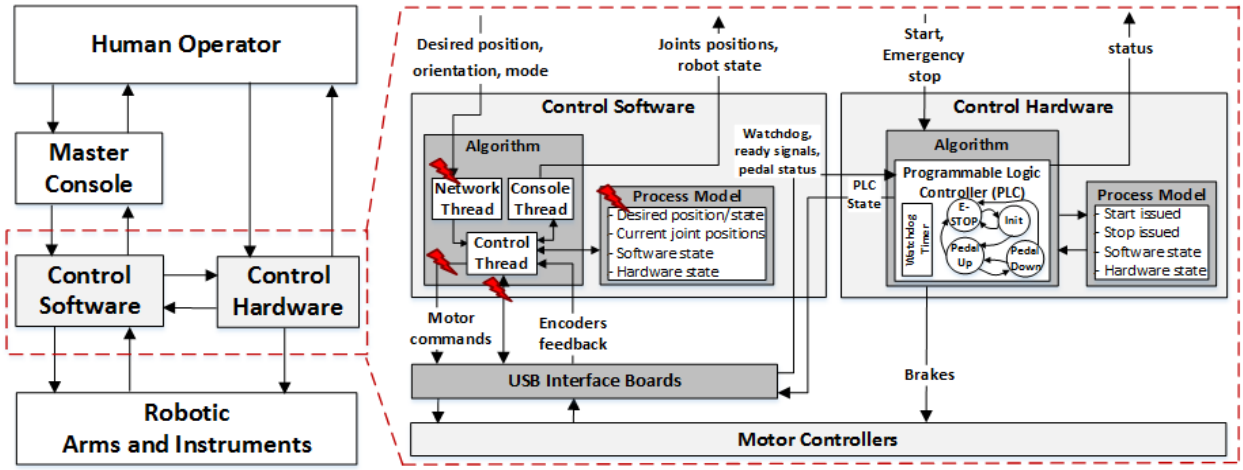


Figure 4.4. (a) Hierarchical control structure of robotic surgical systems, (b) Software and hardware control loops, including control algorithms, process models, control actions, and feedback.

Using STPA, we carefully examined the operation and interactions of components in the software and hardware control loops and found the set of system conditions under which the control actions could possibly be unsafe and lead to hazardous system states (refer to Table 3.1 for the list of potential system hazards and accidents in robotic surgery). For example, in software control loop (in Figure 4.4(b)), we considered any flaws (marked with) in the master console inputs, incorrect feedback from the motor controllers or hardware control, flaws in the process model of software, or output generated by the control algorithm as a potential causal factor. Table 4.1 shows examples of potentially unsafe control actions, safety hazards, and their possible causal factors that we identified for the RAVEN II software and hardware controllers.

As shown in Section 4.3.2, the identified causal factors in combination with the knowledge of software structure provide the scope for performing directed fault-injection experiments. They can define the location within each software module to inject faults, the variables within each function to target, and the conditions to trigger the injections.

4.3.2. Safety Hazard Injection Framework

To evaluate the safety mechanisms of robotic surgical systems, we developed a safety hazard injection framework, which consists of seven modules for retrieving hazard scenarios, generating fault injection campaign, selecting fault injection strategy, conducting fault injection experiment, and collecting data, all in an automated fashion. Figure 4.5 shows the overall architecture of these

Table 4.1. Potential unsafe control actions and causal factors for safety hazards in RAVEN II.

Control Loop	Potentially Unsafe Control Actions			Possible Causal Factors
	Control Action (Type)	Context (System Condition)	Safety Hazards	
Software Control	Motor command (provided)	User desired joint position does not match user desired position	H1-1	- Incorrect console inputs - Faulty control algorithm - Incorrect process model (desired positions, joint positions, runlevel) - Faulty USB communication - Arms/Instruments malfunctions
		User desired joint position is at a large distance from the current joint position (unintended jump)	H1-2	
		Left and right arm end-effector positions are very close (unintended collision)	H2	
		Software State = E-STOP or Software State = Pedal Up, PLC State = Pedal Down	H1 H2	- Missing/incorrect input from PLC - Faulty control algorithm - Incorrect process model (desired positions, joint positions, runlevel) - Missing/incorrect watchdog signal or output to PLC - Faulty USB communication
		Software State = Pedal Down, PLC State = Pedal Up or PLC State = Init	H3	
		Software State = Not E-STOP, PLC State = E-STOP	H3	
	Motor command (not followed)	Software State = Pedal Down or Software State = Init	H3	- Faulty USB communication - Mechanical malfunctions (e.g., broken instruments or cables)
Hardware Control	Brake (provided)	Stop not pressed and Software not stopped/pedal up	H3	- Missing/incorrect watchdog signal or output from software - Faulty USB communication
	Brake (not provided)	Stop pressed or Software is stopped	H1 H2	
	Brake (not followed)		H1 H2	- Mechanical malfunctions (e.g., broken instruments or cables)

modules and how they interface with each other and with the RAVEN II control software and hardware. A detailed description of each module is provided next.

4.3.2.1. Injection Controller

The injection controller is responsible for starting, stopping and automating the fault injection campaign. It communicates with other modules in the safety hazard injection framework through sockets or by direct invocation. In a normal campaign execution, it first accesses the safety hazard scenario library to retrieve the list of hazard scenarios. Second, the controller calls the fault-injection strategies to generate the fault injection parameters that could cause each hazard scenario. Next, it runs the user input generator module and calls the appropriate fault-injector and robotic software to conduct a fault injection experiment. At the end of each injection run, the injection parameters and data are collected and written to the data collector.

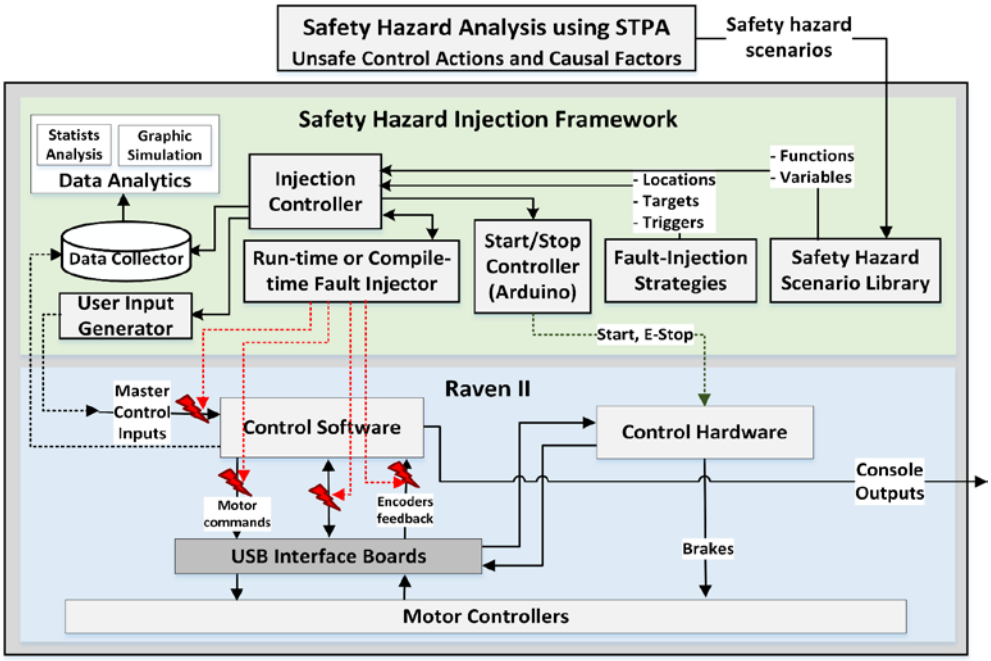


Figure 4.5. Safety hazard injection framework integrated with the RAVEN II surgical platform.

4.3.2.2. Safety Hazard Scenario Library

The safety hazard scenario library contains the safety hazard scenarios identified during the hazard analysis using STPA. Each hazard scenario includes a possible unsafe control action that might happen in the system and a list of potential causal factors. An example unsafe control action would be a motor command is provided by the control software when *there is a mismatch between the software state and hardware (PLC) state of the robot* (rows 4-6 in Table 4.1). *Faulty USB communication* is an example causal factor that might lead to such unsafe control action.

4.3.2.3. Fault-Injection Strategies

Based on the causal factors involved in each safety hazard scenario, the analysis of the RAVEN source code, and software/hardware architecture, the fault-injection strategies module retrieves information on software functions which can most likely result in the hazard scenario, as well as the key variables in those functions and their normal operating ranges. This information is translated to the parameters to be used by the fault-injection engine for simulating potential causal factors and validating whether they lead to the unsafe control or the safety hazards in the system. The fault injection parameters include the *location* in the software function, the *trigger* or condition under which the fault should be injected and the *target* variables to be modified by the injection.

4.3.2.4. Fault Injectors

The fault injectors perform the fault injection during robot operation. The faults are injected with minimum changes to the RAVEN software and hardware, either by replacing the code at the specified *location* with a mutated version that mimics the intended faulty operation (*compile-time fault injection*) [151] or by setting breakpoints at the specified *location* and changing the *target* variables at runtime (runtime fault injection) [151].

Runtime fault injector is implemented by extending the functionality of GDB (GNU Project Debugger for Linux). More specifically, we extend the *breakpoint* feature in GDB to perform fault injection when the desired *trigger* condition is met and then resume the execution of the target program. Runtime fault-injector launches the RAVEN ROS node with GDB Server attached to it, then the extended GDB is run from a remote process and after connecting to the RAVEN node, performs the fault injections. Runtime fault injector has the advantage of performing injections on runtime generated data; however the delay introduced by the runtime breakpoints is not acceptable for modules that have hard real-time requirements. For example, the RAVEN control thread has the hard real-time requirement of one millisecond to perform kinematics calculations and communication with the USB boards [156]. Runtime fault-injection to the control thread introduces small delays, leading to violation of the real-time constraint and failure of kinematics calculations, resulting in unintended robotic instrument vibrations and movements.

Compile-time fault injector is implemented as a module that modifies and recompiles the fault injection conditions into the source code. The main advantage is negligible timing overhead (small compile and build times), which is acceptable for modules with hard real-time requirements. We use compile-time injector to inject faults into the control thread.

4.3.2.5. User Input Generator

User input generator emulates the master console functionality by generating user input packets based on previously collected trajectories of surgical movements made by a human operator and sends them to the RAVEN II control software. It parses the trajectory logs collected from a previous run of the robot and extracts the user input data (desired end-effector positions (*pos_d*), orientations (*ori_d*), mode (*surgeon_mode*)) to construct ITP packets that are sent through a UDP connection to the RAVEN *network_layer* thread with a frequency of 1000 packets per second.

4.3.2.6. Start/Stop Controller

To perform automated fault-injection experiments without manual user intervention, we added a hardware mechanism to automatically start and stop the RAVEN system. This mechanism is added to eliminate the need for pressing the physical start/stop button (see Section 4.2.1) on each run of the robot. We connected the start input of the PLC controller to the output of a relay switch controlled by an Arduino microcontroller [158], which receives software start signals from the Injection Controller. After each injection, the Injection Controller stops the system by shutting down the RAVEN ROS node and user input generator. The next injection gets started by automatically launching the software and sending the start signal to the Arduino relay controller to start the PLC and the RAVEN initialization (homing) process.

4.3.2.7. Data Collection and Analysis

For each fault injection run, the fault injection parameters, user input packets, surgical robot's trajectory, and detected errors are collected using the logging mechanisms provided by ROS (*rostopic*) and are sent to a MySQL server on a remote machine (data collector). This data is later queried for statistical analysis or graphics simulation.

4.3.3. Experimental Results

In our experiments, the identified safety hazard scenarios (unsafe control actions and causal factors) from STPA hazard analysis were used in combination with the knowledge of software code structure to develop the fault-injection strategies. Specifically, we mapped the identified causal factors in each loop of the system control structure to the corresponding software modules and functions that should be targeted for fault injection. In order to increase the possibility of activating the safety hazards in the system, we *manually* defined the variables within each software function to target, the value to inject, and the conditions to trigger the injections. We performed constrained random injections to the variables corresponding to the control action, feedback, process model, and algorithms in every control loop. The injected values were sampled from the within and outside the possible range of variables.

We simulated a total of 45 scenarios (corresponding to the causal factors shown in Table 4.1) by injecting faults into 25 locations within 13 software functions of the network and control threads of the RAVEN II robot, while running a pre-collected trajectory of a surgical movement. Table 4.2

shows examples of scenarios where the faults were manifested in the system. We ran a total of 2,146 fault-injection experiments on the RAVEN robot. However, the majority of the faults (e.g., injected values within the range of variables) were not manifested in the system, or their effect was not logged completely by the data collection process due to system hangs/crashes (e.g., hardware "E-STOP") caused by the faults. For each scenario, we conducted multiple runs (in total 368 fault injections) to get confidence in reproducibility of the manifested/observed system behavior and manually collected the results (see Table 4.2). In each case we analyzed the system behavior both during the homing process (which system is being initialized and user manipulation has not started yet) and after the homing. The third column in the table shows the number of experiments done for each scenario and the last column corresponds to the scenario ID. A complete list of causal scenarios is available at [159].

In this section, we discuss our findings from the conducted fault injection experiments, including the causes for undetected hazards and the hazard scenarios that were mitigated by the safety mechanisms. Section 4.3.4.1 shows representative incident reports from the MAUDE database, which resemble the safety hazard scenarios identified here.

4.3.3.1. Undetected Safety Hazards

Our fault injection experiments covered all the safety hazards and causal factors identified in the STPA process (Table 4.1), but the system safety mechanisms could not detect and mitigate all the simulated hazard scenarios. In what follows we describe the scenarios in which the injected faults led to hazards and were not detected or mitigated by the safety mechanisms in the system.

Unintended Robotic Movement (H1). We found a total of six scenarios where the faults in the console inputs, control algorithm, or the communication between the control software and hardware led to robotic arms/instruments making movements to an unintended position (H1-1) or with an unintended velocity (H1-2).

(i) Out of range values injected permanently into the position, orientation, and foot pedal status inputs received from the master console (in *network_process* function) did not have any impact on the system during the homing process. However, after homing and in "Pedal Down" state, these injections led to kinematics calculations failures, small jumps, or stopping the robot. If the injected values passed the safe limits, movement was stopped by the overdrive detector and E-STOP was raised.

Table 4.2. Example scenarios simulated by fault injection and the observed system behavior.

Potential Causal Factor	Injected Software Fault *Target Function*: Variables [Fault Type, Values]	No.	Observed System Behavior	Hazard	ID
Incorrect console inputs	*network_process*: Position and Orientations [Stuck At Out of Range]	20	During Homing: No impact. After Homing in Pedal Down: IK-failure, small jumps, no movements with no E-STOP, E-STOP.	H1 H3	(i)
	network_process: Foot Pedal Status [Stuck At 0, StuckAt 1]	20	During Homing: No impact. After Homing: Does not start movement if Stuck At 0, No impact if Stuck at 1.		
	network_process: Position and Orientations [Intermittent Out of Range every10, 100, 500 packets]	40	Homing: No impact. After Homing in Pedal Down: IK-failure, No movement, small jumps with no E-STOP, or E-STOP depending on robot configuration.	H1 H3	(ii)
	network_process: Foot Pedal Status [Intermittent 0/1 Flip every 30,100,3000 cycles]	20	Pedal Down: Movement stops or small jumps PLC stops at very high flipping rate (e.g., every other cycle).		
Faulty control algorithm	*TorqueToDAC:* Joints Current Commands [Stuck At -1000]	1	Abrupt jump of both robotic arms, Cables on both left and right arms broke.	H1 H2 H3	(iii)
	stateEstimate: Motor Velocity [Stuck At 0, -1, 1000]	5	During Homing: Unintended rotation, E-STOP. After Homing: No Impact.	H2	(iv)
	stateEstimate: Motor Velocity [Intermittent 0 injection every 100, 3000 cycles]	5	During Homing: Unintended tool movement, hard collision of instrument to the floor. After Homing in Pedal Down: No impact.	H1 H2 H3	
	stateEstimate: Motor Position [Stuck At or Intermittent]	10	Detected and mitigated by (*overdriveDetect*). Raised E-STOP Error and Stopped.	NA	(ix)
Faulty USB communication	*getUSBPacket:* PLC State [Stuck At 0]	12	Homing: Does not start initialization, software assumes hardware is in E-STOP. After Homing: E-STOP, software assumes hardware is in E-STOP, goes to E-STOP, stops sending watchdog, causing hardware to really stop.	H3	(vii)
	getUSBPacket: PLC State [Intermittent 0 injection]	10	Homing: Repeats the homing process over and over again due to synchronization failure of two arms. After Homing: Hardware completely stops or brakes are engaged/disengaged repeatedly.	H2 H3	(v)
	putUSBPacket: Joints Current Commands [Stuck At Random Value]	5	During Homing: No Impact. After Homing: Abrupt jump of robotic arms and cable breaks, Software E-STOP.	H1 H2 H3	(vi)
Incorrect output to PLC	*updateAtmelOutputs:* Output to PLC [Stuck At 0, 1, 3]	16	Does not start the initialization process or stops after homing because hardware goes to E-STOP and gets stuck there.	H3	(viii)

(ii) Intermittent injection of out-of-range values into the master console inputs occasionally caused small instrument jumps or stopping the PLC when the faults were injected at very high frequency (e.g., at every other cycle).

(iii) Injecting a random constant torque value to the joints current commands sent from the control software to the motor controllers (in *TorqueToDAC* function) caused very abrupt jumps of robotic arms, which resulted in the breakage of cables on the arms.

(iv) Faulty estimation of motor velocities by the control algorithm (in *stateEstimate* function) caused unintended rotation and movement of instruments. In one case, upon intermittent injection of zero velocity, the instruments unexpectedly overshot the home position and collided with the surgical field floor during homing process.

(v) Intermittent faulty packets received by the USB interface function (*getUSBPacket*) from the PLC caused the software control to assume that PLC is in "E-STOP" state, while PLC was in "Init" state. During homing process, this fault led the software and PLC to switch back and forth from "Init" to "E-STOP" state, causing failure of synchronization between left and right arms. Therefore, the robotic system got stuck in the initialization process and never moved to "Pedal Up". After homing, depending on the frequency of the intermittent faults, either the robot completely stopped or PLC applied brakes repeatedly to the motors.

(vi) Injecting faults into the packets sent to the motor controllers through the USB interface function (*putUSBPacket*) did not impact the behavior of the system during the homing process, but led to abrupt jumps of robotic arms, resulting in cable breaks. A video recording of this scenario is available at [159].

Unintended Collision or Mechanical Stress (H2). The last four scenarios (iii - vi) discussed above also involved mechanical stress on the robot due to hanging in the homing process, repeating initialization steps, applying brakes over and over again, abrupt jumps of robotic arms, colliding with the surgical field floor, or breaking cables. The robotic system also became unresponsive or unavailable (H3) for almost an hour while repairing each broken cable. Due to the risk of damage to the robot, we repeated these specific injections only a few times.

Unresponsive Robotic System (H3). The majority of undetected safety hazards were due to faults injected in the USB communication or communication between software and PLC (17 scenarios [159]), leading the robotic system to not start the homing process, stop movement,

become unresponsive to the received console packets, or become unavailable due to mechanical issues. Table 4.2 shows examples of these scenarios (vii, viii).

In case of transient or intermittent faults (e.g., in input console packets or USB packets), restarting the system can resolve the E-STOP conditions. However, permanent faults (e.g., a loose or disconnected USB cable causing incorrect information sent from PLC to software, or a DAC malfunction causing incorrect values sent to the motors, simulated as stuck at software faults here) cannot be recovered from even after multiple restarts and by hanging in E-STOP state the robotic system becomes unavailable (H3).

4.3.3.2. Mitigated Safety Hazards

Out of 23 scenarios related to corruption of the console inputs and the control algorithm, only six caused the unintended movements (depending on the robot configuration), collision, or cable damage. All these cases where related to intermittent faults (out of range absolute values) injected into the console inputs (tool positions and orientation or foot pedal) in a periodic manner or to applying constant velocity/torque values to the motors. All other scenarios either did not have any impact (three cases), were detected by the *overdriveDetect* function and mitigated by forcing a hardware “E-STOP” (nine cases) (see scenario ix in Table 4.2), or only caused the system to hang in “Pedal Up” or “E-STOP” with no potential harm (4 cases).

4.3.4. Discussion

The presented approach only assesses the resiliency of the system to the simulated hazards and causal factors considered/identified in the STPA analysis (e.g., H1, H2, and H3 shown in Table 3.1 and causal factors related to software and hardware control loops in Table 4.2). Any other possible safety scenarios which are missed from the hazard analysis process (e.g., a residual fault that might not lead to any unsafe scenarios of Table 4.2 or hazards H1, H2, or H3) will not be created and tested in the system. However, the proposed assessment technique can be used in conjunction with other validation techniques, such as assessing the resiliency of system to the hazards identified from the analysis of real incident data (see Section 4.4) or formal model checking techniques.

4.3.4.1. Related Safety Incidents from FDA MAUDE Database

Table 4.3 shows representative incident reports from the FDA MAUDE database, related to the da Vinci surgical system. In these examples, similar hazard scenarios identified in our STPA analysis (e.g., master console malfunctions and communication failures between the controller and robotic parts) occurred during real robotic procedures. These failures led either to non-intuitive movement of instruments or system errors that could not be cleared even by multiple system restarts.

Table 4.3. Relevant incident reports on da Vinci surgical system from FDA MAUDE database.

Report No. (Year)	Summary Event Description from the Report	Potential Causal Factors (ID in Table 4.2)	Observed Behavior (Hazard)	Patient Impact
2120175 (2011)	During a hysterectomy procedure, the left master controller did not have full control of the maryland bipolar forceps instrument, resulting in non-intuitive motion and causing a small bleed on the patient's uterine tube.	Master console calibration issue (i)	Non-intuitive movement (H2)	Small bleed on patient's uterine tube
2663924 (2012) 2589307 (2012)	Approximately 3.5 hours into a pancreatectomy procedure, multiple instances of non-recoverable system error code #23 was experienced and the surgeon was unable to control the patient side manipulator (psm) arms.	Communication failure between master console and robot (i)	Non-recoverable system error (H3)	Converted to open surgery after 3.5 hours
2721073 (2012)	Prior to starting a da vinci si prostatectomy procedure, the site experienced system error #22003 upon system start up. The site attempted to disable a patient side manipulator (psm) arm and performed a power cycle of the system, but the system error code persisted.	Communication failure between control processor and robot (vii)		Converted to open after port incision under anesthesia

In cases of instruments moving of their own accord or getting stuck due to malfunctions, the consequences may range from minor, where there are just short delays or system resets for troubleshooting the problem, to major, where the instruments may impale or impinge on a bodily structure, causing perforation or bleeding. Tearing or perforation of tissues can cause long term complications and even death. Conversion of procedure to non-robotic approaches is a recovery mechanism to ensure survival of the patients. However, lack of tactile feedback can be a major issue in extracting malfunctioning instruments safely from patient’s body.

Our study demonstrated the value of software-implemented fault injection for simulation of safety hazard scenarios, which might help surgeons recognize complications and act promptly to prevent similar incidents in the future.

4.3.4.2. Vulnerabilities in Safety Mechanisms and Mitigation of Safety Hazards

We discovered the following vulnerabilities in the safety mechanisms of the RAVEN II robot which contributed to the simulated safety hazards:

- Lack of monitoring mechanisms for the initialization (homing) process.
- No safety mechanisms for monitoring the USB board communications.
- No hardware detection mechanisms for monitoring unsafe motor commands.
- No feedback from the motor controllers and brakes to the PLC.

The initial specifications of the RAVEN robot [14] included the requirements for the PLC to monitor the robotic hardware through feedback received from the motors and brakes. However, we found that those monitoring mechanisms are not included in the current implementation of the robot. Also, separate software and hardware mechanisms for monitoring the activities of USB interface boards are needed in the future.

The following robust safety mechanisms had a major role in mitigating safety hazards in the RAVEN II, by preventing unintended movements and possible system damage:

- Robot movements cannot start without a start signal provided by the user.
- PLC engages the brakes upon loss of watchdog ("E-STOP") or foot pedal signals from software ("Pedal Up"); and software only sends the pedal signal to the PLC when the foot pedal is pressed and it is not in "E-STOP" or "Init" state.
- Software checks the status of PLC on every cycle (1 millisecond interval) to immediately follow the state transitions of the robotic hardware.

4.3.4.3. Future Work

Future work involves automated mapping of the control actions, feedback, and process models in different loops of system control structure to the corresponding software functions and variables to target fault injection. This process can be automated by integrating STPA hazard analysis with existing toolchains for requirements analysis and software design. This capability would enable more accurate estimation of the test coverage.

Another topic for future work includes design of tools for monitoring and collection of fault injection results for more accurate estimation of detection coverage. For example, at the end of a fault injection experiment, a system hang or crash (e.g., hardware emergency stops) might occur either due to the injected fault or naturally because of other calibration issues of the robotic system.

In our experiments we manually observed the behavior of the system in several runs of the same experiments and distinguished these two scenarios from each other. A more accurate estimation of detection coverage requires developing either higher fidelity simulators that enable emulating the whole system behavior in software or tools for automatically distinguishing such scenarios from each other.

4.4. Simulation of Safety Hazards for Safety Training

Motivated by the idea of including safety-critical events in simulation-based training of system operators, we developed a platform for simulation of the safety hazard scenarios extracted from the analysis of real FDA adverse events. As shown in Figure 4.6, this simulation platform is built by integration of the following components:

- A robotic surgical simulator based on the control software of the RAVEN II robot that enables running of robot control software without the robotic hardware by modeling the functionality of the robotic hardware and mechanical components and visualization of robot behavior in a 3D environment.
- Our existing toolset for systems-theoretic causal analysis of adverse events (see Section 3.3), which is used to identify commonly observed hazardous system states and their causal factors from the FDA MAUDE data to populate a library of safety hazard scenarios that define the location and type of faults and the conditions under which they should be injected into the software (see Section 4.3.2.2).
- The safety hazard injection framework (presented in Section 4.3.2) that mimics hazard scenarios (involving robotic software and hardware, human operators, and the human-computer interface) by automatically injecting faults into different modules of the robot control software.

The simulation platform provides the flexibility of either using real human input from a master console (e.g., an Omni haptic device) or replaying previously-collected trajectories from real surgical tasks using a master console emulator (see Section 4.3.2.5).

We simulated the RAVEN II hardware by developing a software module that mimics the dynamical behavior of the real robotic actuators by modeling the MAXON RE30 and RE40 DC motors used by the RAVEN II robot [15] as first-order systems with different time constants. A

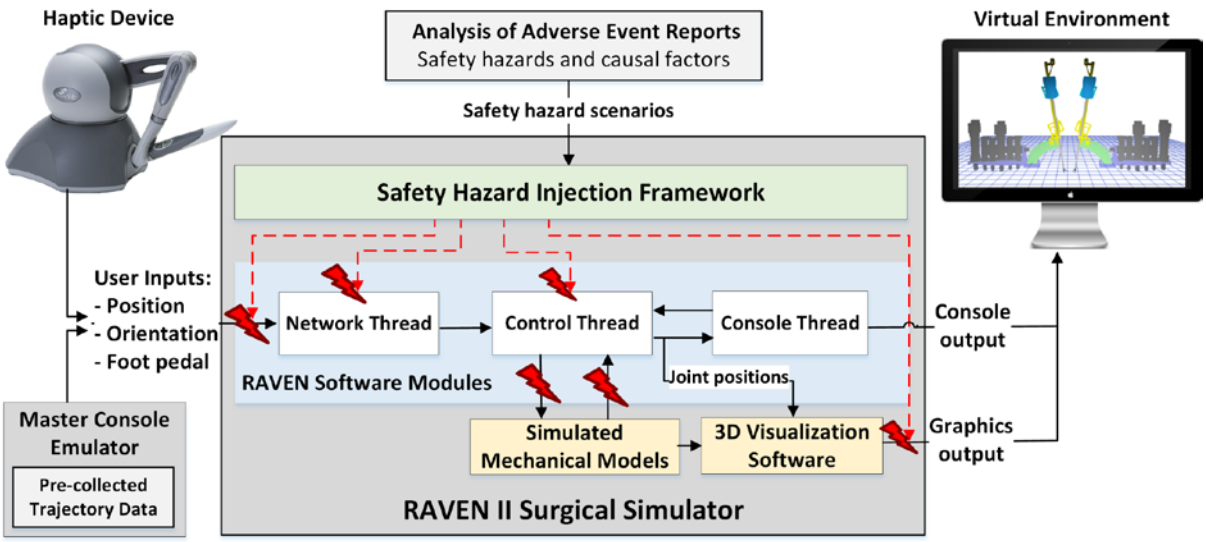


Figure 4.6. RAVEN II surgical simulator integrated with the safety hazard injection framework.

3D virtual environment based on C++ OpenGL pipeline and CAD models of robot mechanical components was created to animate the movements of robotic arms and instruments.

4.4.1. Library of Safety Hazard Scenarios

In our proof of concept study, we focused on the safety hazard scenarios involving device and instrument malfunctions and inadequate operational practices that contributed to catastrophic events or interruptions during robotic procedures (examples shown in Table 4.4 and Table 4.5).

Table 4.4 shows samples of representative adverse events in which malfunctions of master manipulators (the inputs at the master console that the surgeon uses to send control commands to the robotic instruments) or motor encoders and potentiometers (the sensors at the patient side that collect measurements from robotic arms and instruments as feedback to the master console) or other device-related failures led the safety processor to stop the system and raise system errors during a procedure. The majority of system errors in Table 4.4 could not be resolved even by multiple system restarts and eventually led the surgical team to convert the procedure or abort and reschedule it to a later date.

Table 4.5 shows example events in which inadequate operator actions, due to deficiencies of the human-machine interface or lack of training, contributed to unexpected device operation and led to adverse impact on patients. None of the events in Table 4.5 resulted in a system error. The

full descriptions of example MAUDE reports shown here are accessible through searching the report numbers in the online FDA MAUDE database [64].

Table 4.4. Example adverse event reports that involved device and instrument malfunctions.

Report No. (Year)	Summary Description	Malfunction Type	Procedure Outcome
1006071 (2008)	- Recurring system errors #201 and #264, even after multiple restarts. - Errors due to voltage tracking faults and put the system in a recoverable safe state.	Master tool manipulators	Converted after 2 hours
3283230 (2013)	- Master tool manipulator arm was sluggish and could not control the robotic arms. - System error #22580 due to out-of-range hardware voltage level. - Multiple system restarts did not resolve the issue.		Aborted post anesthesia
3093014 (2013)	- Recurring error #23000, even after emergency power off & restart. - System error caused because the angular positions of one or more robotic joints on a manipulator as measured by the primary sensor (encoder) and secondary sensor (potentiometer) were out of range or in disagreement.	Joint sensors (Potentiometer or encoder)	Aborted post anesthesia and port incision
2916352 (2012)	- Recurring system error #23008, even after emergency power off & restart. - Recoverable errors caused because the angular positions of robotic joints as measured by the primary sensor (encoder) and secondary sensor (potentiometer) were out of range or in disagreement.		Converted after port incision
2014 (3620041)	- Non-recoverable error #23013 on patient side manipulator - Multiple system restarts to recover from error but unsuccessful		Converted to open surgery

Table 4.5. Example adverse event reports that involved improper operational practices.

Report No. (Year)	Summary Description	Improper Operational Practices	Procedure Outcome
921167 (2007)	- Patient-side manipulator dropped suddenly. - Scissors instrument bumped into uterus.	Surgeon removed his/her hands from master manipulators before removing his/her head from console viewer (keeping head in the console viewer keeps the robot engaged)	Pierced patient's uterus
1570678 (2009)	- Endoscopic camera manipulator difficult to move. - Master tool manipulator drifted when released.		Punctured patient's uterus
1961862 (2010)	- Instrument moved to guided tool change mode, moved slightly forward, and bumped into colon.		Injury to patient's colon
2644122 (2012)	- Uncontrolled movement of master manipulators.		Damaged abdominal wall
2636117 (2012)	- Limited range of motion and drift while master tool manipulators were used, even after system restart.		Aborted after 1.75 hours
2476271 (2012)	- Monopolar energy was released when bipolar instrument was used.	Improper connection of bipolar instrument to electrosurgical unit	Injury to patient's bowel
3024317 (2013)			
2494890 (2012)	- Arcing from bipolar instrument when cautery energy was not being applied.		Small burn on diaphragm

Table 4.6 summarizes three safety hazard scenarios commonly observed in the MAUDE data along with their corresponding unsafe control actions and example possible causal factors. We simulated these hazard scenarios in the same way they were reported in the MAUDE database by injecting faults into the RAVEN control software. Note that possible causes of hazards may include accidental faults in robotic hardware or software, and human operator errors, or potential malicious tampering with the system as shown in Figure 3.3. A more detailed discussion on the security related causes of safety hazards is the subject of Chapter 5. For each hazard scenario, Table 4.6 shows the methods for recreating it in the simulation platform as well as its potential impact on

Table 4.6. Example safety hazard scenarios and corresponding methods to recreate them in the simulator.

Safety Hazard Scenario (Outcome)	**Unsafe Control Action (Control Loop No.)**	**Possible Causal Factors (Accidental Failures or *Malicious Attacks*)**	**RAVEN II Simulation**		**Patient Impact (Clinical Scenarios for Safety Training) [MAUDE Report #]**
			Target Software Module	**Target Variables**	
H-3: System temporarily unavailable **(Recoverable System Error)**	A user command is provided but not followed by the robot (3)	Improper operator actions or master manipulator malfunctions *Corruption of user inputs by Man-in-the-middle (MITM) attack*	Network-Layer Thread (*network_layer*)	User-desired -Position -Orientation -Grasper angle -Foot pedal	Restart the system [3293519] Troubleshoot error Contact manufacturer
H-3: System permanently unavailable **(Non-Recoverable System Error)**	A motor command is provided by the robot control, but it is not followed by the motors. (6)	Sensor (encoder) malfunctions *Corruption of sensor data by getting unauthorized access to the system*	Control Thread (*get_USB_packet*)	USB Board -numbers -address -returned status	Convert procedure [2663924] Reschedule [3275500] Report to manufacturer
		Actuator (motor controller) malfunctions *Corruption of motor commands sent to hardware by getting unauthorized access to the system*	Control Thread (*put_USB_packet*)	USB Board -numbers -address -returned status	Puncture of artery [1590517] Bleeding of uterine tube [2120175]
H-1 and H-2 Unintended movement of robotic arms **(Sudden Jump)**	A command is provided by the robot control to motors while the calculated next position is at large distance (big jump) from current position. (6)			Commands to robotic joints	

Full descriptions of example MAUDE reports shown here are accessible through searching the report numbers in the online FDA MAUDE database [64].

patients (based on example incidents reported in the MAUDE database) to represent the clinical scenarios on which the robotic surgeons can be trained.

4.4.2. Experimental Results

To regenerate the safety hazard scenarios shown in Table 4.6, we injected a total of 5,500 faults into the network and control threads on the surgical simulator and 110 faults on the actual RAVEN II robot, while running a pre-collected trajectory of a simple surgical movement. The following sections describe simulation of each hazard scenario by recreating example of their corresponding causal factors.

4.4.2.1. Master Manipulator Malfunctions

The system errors due to malfunctions of master manipulators (the first scenario in Table 4.6) were simulated by injecting random faults into the ITP packets received by the *network-layer* thread in the RAVEN control software. The fault-injection framework targeted the position, orientation, grasper angle, and foot pedal variables in the packet data structure by modifying their values into random values outside the range of possible values that the variables take. Since the ITP packets are sent in an incremental motion scheme, modifying the variables of a few packets to values inside the range of possible values did not have any impact, but intermittent injection of out-of-range values to the position, orientation, and grasper angle variables for an extended period caused the kinematics calculations to fail. The RAVEN II control software detects such failures by observing an over-the-limit electrical current command being sent from the software to the digital to analog converters (DAC) on the motor controllers and raises an E-STOP software error, leading the hardware watchdog timer to move the RAVEN hardware to an E-STOP safe state. The E-STOP error can only be resolved by restarting the system, resembling a recoverable system error scenario. Depending on the length of the faulty packets (e.g., if the master manipulator malfunction is permanent), the E-STOP error cannot be recovered from, even with multiple restarts, which is similar to a non-recoverable system error scenario.

For this hazard scenario, we performed 30 injections into the network-layer thread on the actual robot, from which 22 were manifested in the system. Faults injected into the foot pedal variable or and those with values within the range of possible values, did not manifest in our experiments.

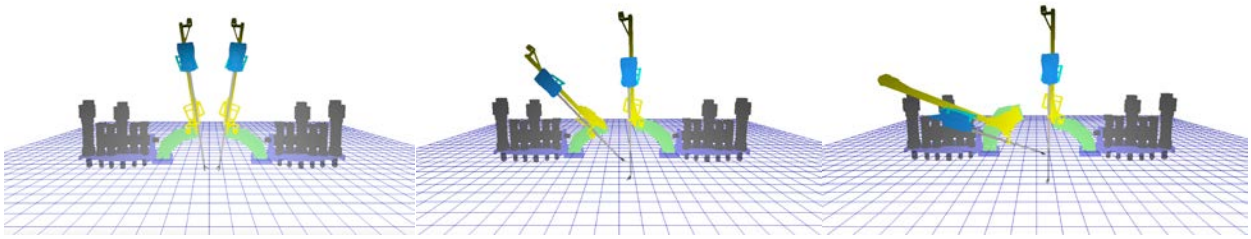


Figure 4.7. Visualization of a safety hazard scenario in the virtual environment: The left robotic arm makes a sudden jump because of a faulty packet sent from control software to the motors.

4.4.2.2. Sensor Malfunctions

We simulated non-recoverable system errors due to permanent or intermittent sensor malfunctions (the second scenario in Table 4.6) by injecting 64 faults into different locations within the USB interface function responsible for communicating packets from motor controllers to the control thread through USB interface boards (*get_USB_packet*). 61/64 of the faults injected into *get_USB_packet* function after the initialization phase were manifested, and the rest (injected during the homing process) did not manifest as safety hazards.

Corruption of the number of available USB boards, indices for accessing USB boards (their address), and packets read from the USB boards (returned status) caused the RAVEN watchdog processor to detect an error and put the system in a non-recoverable E-STOP state, from which we could not recover by simply pressing the physical restart button. Only a complete restart of both the RAVEN software and robotic hardware resolved the issue.

4.4.2.3. Improper Human Operation or Actuator Malfunctions

To simulate unintended instrument movements and sudden jumps of robotic arms due to actuator malfunctions or improper human operations (the third scenario in Table 4.6), we injected twelve faults into the USB board variables and four faults into the robotic joint command variables in the USB interface function (*put_USB_packet*). Ten out of twelve of the faults injected *put_USB_packet* function after the homing process were manifested as non-recoverable system errors (similar to previous hazard scenario). The injections into the robotic joint commands in the USB packets sent from the control thread to the interface boards caused abrupt jumps of the robotic arms, leading the RAVEN software and hardware to stop. Since abrupt jumps of robotic arms

could potentially lead to system damage, we repeated these injections only four times. Figure 4.7 shows the visualization of this safety hazard scenario in the 3D virtual environment.

4.5. Related Work

4.5.1. Fault-Injection for Dependability Evaluation

Software implemented fault injection (SWIFI) [151], [152] has been used for evaluating the dependability of different computing systems, including operating systems [160], [161], smart power grids [162], and SaS cloud platforms [163]. International safety standards, such as NASA Software Safety Guidebook [149] and functional safety standard for automobiles (ISO 26262) [148], recommend using fault-injection for validation of safety-critical software [150]. However, medical devices safety standard (ISO 14971 [20]) do not consider fault-injection testing for validation of medical software [164]. Only one study showed the use of software simulation fault injection for testing the UML model of software for a pacemaker device [165]. In our work, we showed the feasibility of using software fault-injection for empirical assessment of safety mechanisms in surgical robots and validating the safety requirements identified using systems-theoretic hazard analysis techniques such as STPA.

4.5.2. Systems-Theoretic Safety Analysis

STPA has been used for hazard analysis and safety-based design in safety-critical domains such as aviation [55], medical devices [57], and automotive systems [58], [59]. Most previous studies used STPA only to derive the high-level safety constraints and identify the unsafe interactions that should be eliminated or controlled during the design process. In [58], [59] the authors use the results of the STPA analysis to manually develop formal specification of the system safety requirements using temporal logic (e.g., LTL and CTL) and verify software against the safety requirements using model checking tools. The limitation of these works are that (i) manual interventions are needed for both mapping the safety properties identified by STPA to temporal logic and formally modeling the software code in the input language of target model checker, and (ii) they cannot easily scale to the size and complexity of real-world systems due to state explosion (a well-known problem in formal software verification [166]). Our approach also involves some manual intervention for development of safety hazard library (fault-injection targets) based on the

results of the hazard/ accident analysis and mapping the functionality and interactions of the components in the system control structure to the corresponding software modules and functions. However, this process can be automated by integrating STPA hazard analysis with existing toolchains for requirements analysis and software design. For example, [167], [168] recently proposed an extension for the UML/SysML modeling tool Sparx Systems Enterprise Architect, where the elements of the system control structure and STPA results (unsafe control actions and causal factors) are linked to the design models of the system and can later be traced in the software code automatically generated from the UML/SysML models.

4.5.3. Simulation-Based Safety Training

Multiple studies have shown that simulation can be effectively used in training to improve skill levels of robotic surgeons. There are already several surgical simulators, training centers, and validated curricula for robotic surgery [169] – [176]. However, the emphasis has been only on improving surgical skills and not on handling safety-critical events and responding to technical problems. To the best of our knowledge, our work is the first to include the adverse events or safety-critical events (including common device failures and operational mistakes) as scenarios in safety training of robotic surgeons.

4.6. Conclusions

We presented a framework for simulation of safety hazard scenarios identified using systems-theoretic hazard analysis or constructed based on causal analysis of real adverse events from the FDA databases. We demonstrated the feasibility of using this framework for validating the robustness of the system safety mechanisms during design and implementation phases and for evaluating human operator performance and response to safety hazards in simulation-based training.

The proposed framework was evaluated using the RAVEN II system, an open-source platform for research in telerobotic surgery. A software-implemented fault injection framework was developed to simulate the hazard scenarios by inserting faults at critical locations within robot control software, which were identified based on systems-theoretic hazard analysis using STPA.

We also developed a robotic surgical simulator augmented with the fault-injection framework to simulate realistic safety hazard scenarios commonly reported for robotic surgical systems in a 3D virtual environment. The proposed surgical simulator can be used for safety training of robotic surgeons to prepare them for handling common types of adverse events experienced during procedures.

More broadly, software-implemented fault injection directed by the systems theoretic hazard analysis enabled us to: (i) identify the safety hazard scenarios and determine their potential causes; (ii) trace propagation of faults in the system and discover the vulnerabilities in system safety mechanisms; (iii) determine strategic placement of new detectors that can mitigate the propagation of causal factors into safety hazards; and (iv) provide useful feedback to the system developers on how to improve the safety mechanisms in the next-generation devices.

CHAPTER 5
SAFETY-CRITICAL CYBER-PHYSICAL ATTACKS: PREEMPTIVE DETECTION AND MITIGATION

5.1. Overview

The increasing complexity and connectivity of medical cyber-physical systems and their rapidly growing deployment in hospitals and widespread use in a variety of clinical settings, make their resiliency (i.e., their ability to maintain an acceptable level of safe operation) despite both accidental faults and malicious attacks a challenging task. Many recent reports indicate the existence of vulnerabilities in the configuration of hospital networks [179], [180], the third party networks (e.g., laboratories and pharmacies) [181], [182], the devices used by the hospital employees (healthcare professionals and technicians) [183], [184], and unpatched medical devices [185], that may allow attackers to penetrate into hospital networks and potentially gain unauthorized access to the safety-critical medical devices.

In this chapter, we introduce a family of malicious attacks on medical cyber-physical systems that can lead to violation of safety constraints, by significantly disrupting system operations or harming patients. We specifically focus on *teleoperated* surgical robots as an example of the safety-critical medical devices that are envisioned to be used in the future for operation in remote and extreme environments, such as disaster-stricken areas, battlefields, and outer space [186].

Past studies have emphasized the importance of security in teleoperated surgical systems [187] – [191]. Studies [189] – [191] demonstrated the importance of denial of service (DOS) and man-in-the-middle (MITM) attacks that compromise the network communication between the surgeon's console and the robot. To the best of our knowledge, no previous work has discussed the possibility of compromising the control systems of surgical robots. It is usually assumed that getting access to the robot control system is unlikely.

This chapter contains material from the works [177], [178], coauthored with D. Chen, X. Li, A. Lewis, P. M. Cao, Z. Kalbarczyk, R. K. Iyer, and T. Kesavadas, copyrighted by IEEE.

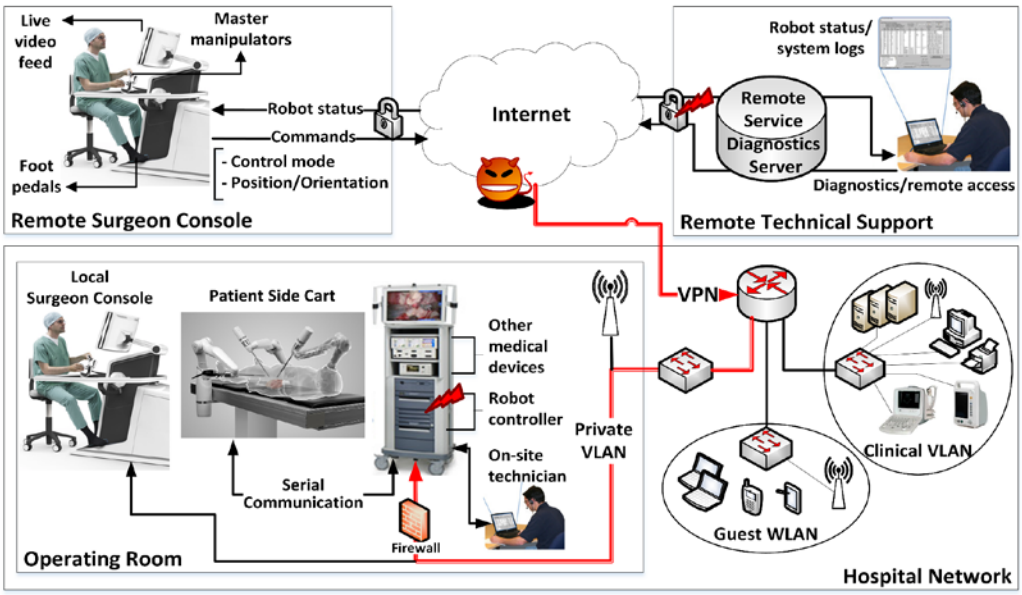


Figure 5.1. Typical network structure in a hospital, hosting a telerobotic surgical system and possible entry points for unauthorized access to the robot. (The images are adapted from [192], [193].)

We demonstrate that surgical robots are potential targets for malicious software that can be carefully installed by the attacker on the robot *control system* to strategically inject faults into the system at a critical operational state during surgery, in order to cause non-deterministic failures, and hence endanger the success of the robotic procedure and threaten the lives of patients.

The attack is deployed via a self-triggered malware with embedded: (i) *logging* mechanisms for collecting and analyzing measurements from the surgical robot in order to identify the critical states (triggers for injection) and (ii) *fault-injection* mechanisms for inserting malicious commands into the robot control software. The deployment of the malware presumes that the attacker has penetrated the hospital network by exploiting vulnerabilities in the underlying network infrastructure and has obtained access to the robot control system through stolen credentials or exploiting one of the zero-day remote code execution vulnerabilities (shown in Figure 5.1 and discussed later in Section 5.2.4) to install the malware.

The type of attack we construct here modifies the motor control commands sent from the control software to the physical robot, while preserving its legitimate format, making this type of attack difficult to detect without understanding the semantics of commands, i.e., dynamics of the robot's control equations. To detect and mitigate such attacks, we have developed a model-based analysis framework that can estimate the consequences of control commands through real-time computation of the robot's dynamics and can preemptively determine if a command is unsafe

before the actual execution of the command can progress and its adverse consequences manifest in the physical system. We experimentally validated the proposed detection scheme using two real attack scenarios involving injection of unintended user inputs and unintended control motor torque commands to the robot control software.

We illustrate the attacks by implementing a prototype of the malware targeting the RAVEN II robot. We use the RAVEN robot as our experimental platform for several reasons: (i) it contains the typical control and safety mechanisms used in state-of-the-art robotic surgical systems, (ii) it is a platform indicative of the next-generation of teleoperated surgical robots with both remote operability and networking features, and (iii) it is accessible for demonstrating security attacks and studying their impact without the need to interrupt real surgical procedures or risk of harming patients.

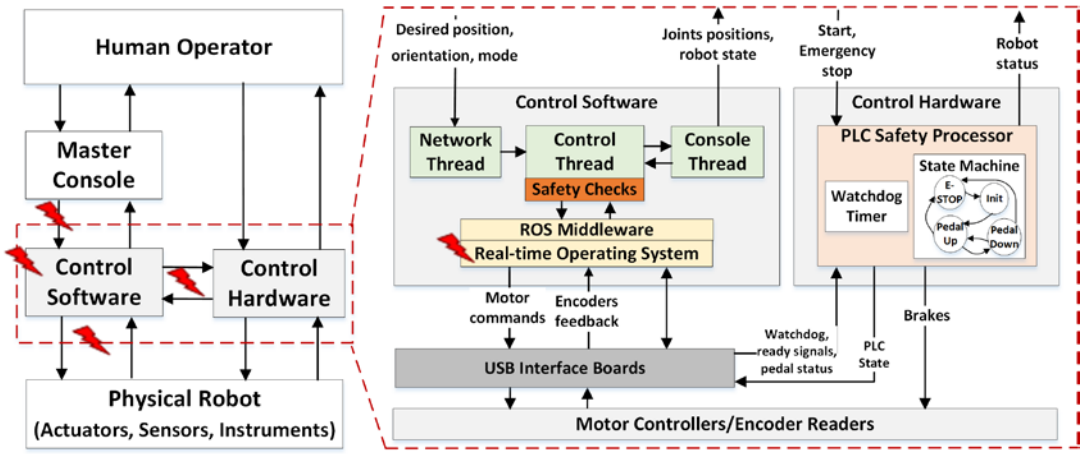
The cyber-physical attack scenarios presented here have the following important characteristics that complicate their detection and diagnosis:

(i) Attacks exploit the TOCTTOU (Time of Check To Time of Use) vulnerability [194], [195], from the time the safety checks on the motor commands are done to the times the commands are written to the file system, are sent to the USB boards, and their actual execution on the robot.

(ii) Attacks are initiated in the cyber domain by modifying the control commands while preserving the legitimate format and syntax, i.e., no changes are made to the control flow (in terms of the sequence of the functional blocks invoked) and to the performance of the target program (preserving the real-time constrains of the robot control software).

(iii) Attacks directly result in catastrophic consequences in the physical domain (e.g, abrupt jumps of the robotic arms), causing damage to the robot or harm to the patient. Thus, they are hard to distinguish from incidents caused by system malfunctions or human induced accidental failures.

The stealthy nature of these attacks and their resemblance to commonly observed accidental failures in surgical robots makes them more favorable to attackers, compared to other attack scenarios that simply kill the robot or make it unavailable during surgery. If deployed on wide scale, such attacks could cause major disruption and damage to surgical facilities and cause financial or legal impacts.



(a) (b)

Figure 5.2. (a) Control structure and (b) Software and hardware control loops in the RAVEN II robot.

We also present two possible scenarios of easier attacks that compromise the availability of the robot during the procedure by exploiting the vulnerabilities in open-source Robotic Operating System (ROS) and remote diagnostic services used by the commercial surgical robots (da Vinci surgical system) in Section 5.4.

The dynamic-model based analysis framework presented here can be generalized for analysis, detection, and mitigation of safety hazards caused by either accidental failures or malicious attacks.

5.2. Targeted Attacks on the Control System of Surgical Robots

Our previous study in Section 4.3 revealed several vulnerabilities in the safety mechanisms of the RAVEN II robot. In this chapter, we show that malicious parties can exploit such vulnerabilities to perform cyber-physical attacks that are difficult to be detected without modeling the dynamic equations of the robot's control equations.

The attacks exploit the dynamic loading feature for system libraries in the underlying Linux operating system and the vulunerabilities in the RAVEN II software-hardware interface to inject malicious actions at different layers of the robot control structure (shown in Figure 5.2). These attacks can cause a variety of adverse impacts on the robot functionality and the patient and these impacts are potentially difficult to distinguish from unexpected failures. Table 5.1 summarizes variants of these attacks, categorized by the target layer in the control structure (shown in Figure 5.2(a)), the target system library, the type of malicious action, and their anticipated impact

Table 5.1. Variants of attacks targeting different layers of robot control structure.

	Target Layer	Target System Library	Malicious Action	Observed Impact
A	Master Console to Control Software	Socket communication (bind, received_from)	Change -port number -packet content	Hijack trajectory Unwanted state (E-STOP)
	Control Software	Math (sin, cos)	Add drift to -output -input	Unwanted state (IK-fail)
	Control Software to Hardware	Interface (read, write)	Change -robot state in PLC	Homing Failure
B	Control Software to Physical Robot		Change -motor commands -encoder feedback	Abrupt Jump Unwanted state (E-STOP)

on the system. We specifically focus on two attack scenarios that cannot be preemptively detected and mitigated by the existing safety mechanisms in the RAVEN II robot:

A. **Injection of unintended user inputs** after they are received by the control software from master console. These attacks either cause hijacking the control of the robot by performing an action that was not initiated by the operator or lead to unintended jumps and unwanted halt states.

B. **Injection of unintended motor torque commands** after the they have passed the safety checks in the control software and before transmission to the USB interface boards and motor controllers on the physical robot. These attacks lead to unintended moves and abrupt jumps of the robot or to unwanted halt states.

We exemplify the attacks by deploying attack scenario B (described above) on the RAVEN II robot. We used a desktop computer running the RAVEN II software on top of ROS Indigo and Linux Ubuntu 14.04 LTS with SMP Preempt Real-time kernel. The machine contained an Intel Core i5 CPU@2.90 GHz and 8GB of RAM. The malicious code was implemented using bash, Python scripts, and ROS commands and was executed in the user space (as discussed in Section 5.2.1, the attack can be deployed by either by compromising a user or gaining an unauthorized root privilege).

5.2.1. Attack Model

We focus on the steps taken *after* the attacker has obtained remote access to a robot control system on a hospital's network. Figure 5.1 shows the structure of a hospital's network hosting a teleoperated surgical robot. Although, the current generation of commercial surgical robots are not

used for remote or telesurgery applications, based on the evidence presented in the related work and publicly-available documents, the typical configuration of a teleoperated robot on the hospital network can be envisioned as follows:

- The surgical robot (including the local master console, the robot controller, and other related medical devices in the operating room) are connected to a private VLAN (Virtual LAN) within the hospital network. The computer hosting the robot control software is protected by an internal firewall, restricting access only to a few ports used for teleoperation and remote diagnosis [180], [196].
- The robotic system can be operated either by surgeons at remote surgical sites by sending the surgical commands through the Internet or by the local surgeon through master consoles connected to the local network.
- Both the on-site and remote technicians can have access to the robot control system for retrieving diagnostic logs and troubleshooting robot problems through either wireless or wired connections.
- The remote master consoles or technicians can only access the robot control system through a secure connection (e.g., VPN connection to the hospital) and after getting access to the private VLAN [196].

As discussed before, the attacker can gain unauthorized access to the hospital by exploiting weaknesses such as vulnerable services, unpatched medical devices, stolen credentials, or insider attacks to penetrate the hospital network. Once in the hospital network, the attacker can move laterally across devices within the hospital, steal additional credentials and discover vulnerabilities until the target robot control system is located and penetrated. The attacks discovered by TrapX Security, Inc. [185], the Stuxnet attack [197], and the discovered vulnerability in the firewall of a commercial robot [180] serve as examples of how these penetration attacks could be performed. Table 5.2 shows the common entry points exploited in recent real attacks detected on hospital networks. The purpose here is to assert that access to the robot control system in present day environments is not only feasible but quite probable.

After getting access to the robot, the intention of the attacker is to remain on the target system without being detected for as long as possible in order to (i) collect data from the system, (ii) analyze the collected data to create an operational profile of the robot and determine the best time for activating the attack, and (iii) trigger the attack at the desired critical time.

Table 5.2. Potential entry points onto a hospital network and examples of real attacks.

Attack Entry Points	Description	Examples of Real Attacks and Detected Vulnerabilities	Ref (Year)
Third-party networks	Hospital networks are often connected to third party laboratories, pharmacies, and vendor networks that, if compromised, can let the data breaches or penetrations into the hospital networks as well.	Two medical centers and more than 3.9 million individuals were affected by a data breach through a third party portal/personal health record platform.	[181] [182] (2015)
Computers used by physicians, nurses, or technicians	Physicians, nurses, and vendor support technicians usually have remote access to the hospital network. The computers they use to access the hospital network could be compromised through credential stealing, virus, and malware.	Email phishing attack compromised personal information of 3,300 patients.	[183] (2014)
Vulnerable office devices	Office devices such as network attached desktops, printers, faxes, scanners, and security cameras with default or weak passwords or vulnerable firmware could be an easy entry point.	Default username/passwords for the multi-function printers and security cameras could be used for access to other devices on the hospital.	[184] (2014)
Vulnerable or incorrectly-configured firewalls, access points, or gateways	Incorrect configurations in the Wifi access points or gateway machines could expose vulnerabilities or leak information, such as device ID or hospital network layout, to the public.	Incorrect configuration of a gateway computer leaked critical information that made it possible for attackers to locate vulnerable devices within the hospital's network.	[179] [180] (2014)
Vulnerable medical devices	Medical devices, such as X-ray systems, infusion pumps, blood gas analyzers, which are connected to the hospital network often have default/weak password or have unpatched software/firmware, which can be compromised.	Three real-world attacks were detected by TrapX where a blood gas analyzer, a PAC system, and an X-ray machine were hijacked to open backdoors in hospital networks.	[185] (2015)

We assume the attacker does not have access to the source code or internal design of the robot. The attacker gathers information about the system configuration and potential vulnerabilities of the robot through publicly available documents (e.g., previous publications on vulnerabilities of the RAVEN II robot) or through a vulnerability discovery process consisting of targeted probing and fuzzing.

There are specifically two pieces of information that the attacker must have about the RAVEN in order to perform a successful attack:

(i) the state machine representing robot operations (see Figure 4.2(b)), and

(ii) a side channel that can be used to extract the current state of the robot in order to determine the best time to trigger an attack.

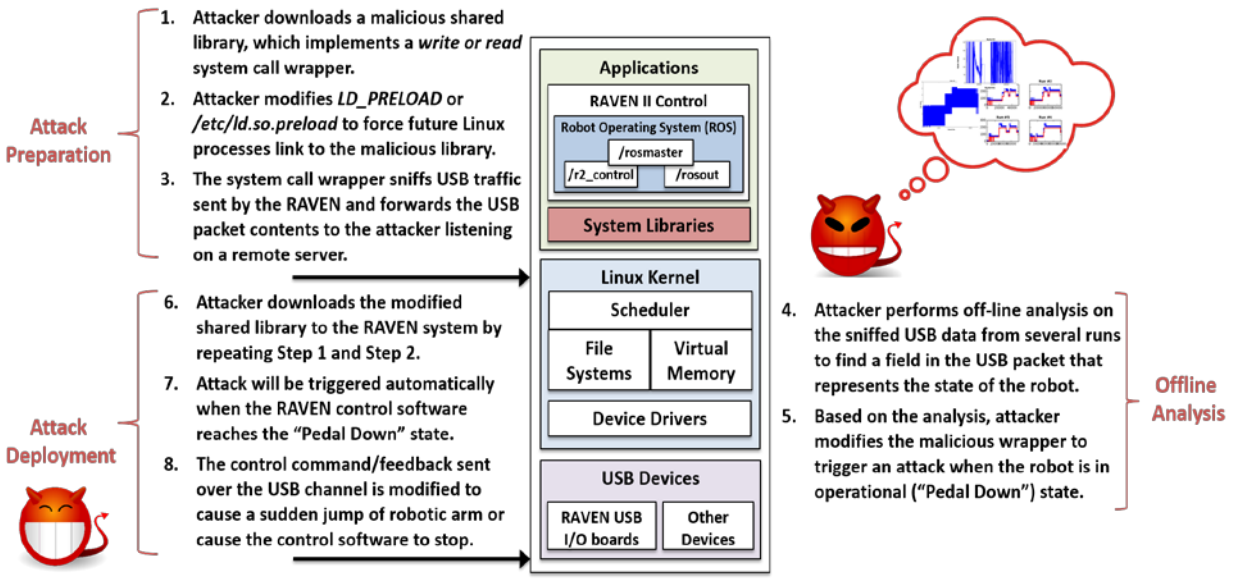


Figure 5.3. Attack scenario B (injection of unintended motor torque commands) in RAVEN II robot.

5.2.2. Attack Description

In the attack scenario illustrated in Figure 5.3, an attacker (who penetrated the RAVEN control system) first eavesdrops (intercepts) on the USB communication between the RAVEN control software and the USB I/O boards. The intercepted packets are analyzed offline to extract the state information of the surgical robot, i.e., determine the state of the robot according to the operational state machine depicted in Figure 4.2(b). The extracted data is then used to build a malware for triggering (injecting) an attack at a critical time during the robot's operation, i.e., when the robot is operating in the "Pedal Down" state.

Figure 5.3 describes the steps to execute the attack on a RAVEN II robot, which are grouped into three phases: *Attack Preparation Phase*, *Analysis Phase*, and *Attack Deployment Phase*. The Attack Preparation Phase and the Offline Analysis Phase need to be performed only once to obtain the information necessary to design and implement the final malware capable of triggering an attack when the robot is most vulnerable. The details of each phase are described next.

5.2.2.1. Attack-Preparation Phase

The goal of the Attack-Preparation phase is to eavesdrop on the communication between the RAVEN control software and the USB I/O boards and send that information to the attacker for offline analysis. This is achieved by (i) downloading and installing a malicious shared library on

the RAVEN control system, (ii) forcing processes on the system to link to the malicious shared library, and (iii) logging the RAVEN USB communication and forwarding it to the attacker on a remote server using UDP packets.

In a Linux system most programs do not communicate directly with the kernel. Instead, the program invokes a function in a runtime library (e.g., *libc*), which performs the necessary preparation of the arguments and then triggers the corresponding system call. When a program starts, the runtime linker searches the default path to find the runtime library to be linked. If an environment variable *LD_PRELOAD* or the directory */etc/ld.so.preload* is defined in the system, then the linking process is forced to first search, load, and link to the library object in the path pointed by the *LD_PRELOAD* or */etc/ld.so.preload* [198]. If the alternative library object has a function with the same name as function defined in the original runtime library (e.g., *read* or *write*), the alternative library's function will be called. This allows the alternative library to "wrap" the runtime library function and intercept system calls. The alternative library function can call the original system call, not call it, or do some malicious task before calling it. This approach has been used by several rootkits to hide their operations, such as Jynx [199] and Azazel [200].

In implementing the attack scenario B, we exploited the Linux dynamic linking feature to install malicious system call wrappers for the *read* and *write* system calls in order to eavesdrop on: (i) commands sent to the robot motor controllers and (ii) the feedback received from the motor encoders and from the PLC through USB. An attacker with the user privilege, can add the *LD_PRELOAD* environment variable to user's startup profile (e.g., *.bshrc*), so all future terminals started by this user will have the *LD_PRELOAD* environment variable set to point to the malicious shared library. The attacker with root privilege can add the path to the malicious shared library to */etc/ld.so.preload*, so that new processes started on the system by any user link with the malicious shared library. This means that when any future process makes a *read* or *write* system call, the system call wrapper in the malicious shared library will be called.

Figure 5.4(a) shows a sample *usb_write* function that uses the *write* system call to write to a USB device. Each file descriptor (*fd*) is associated with a target USB I/O board (on left or right robotic arm) and the *buf* parameter contains the packet containing the motor commands sent to that USB board.

Since the system call wrapper is applied to all programs that are linked to the malicious shared library, we need a mechanism to detect when the system call is called by the RAVEN process and

Raven Code

```
int usb_write(int id, void *buffer, size_t len)
{
    // write to board
    int ret = write(boardFPs[id], buffer, len);

    if (ret < 0)
      ret = -errno;
    return ret;
}
```

Libc Code

```
ssize_t
__libc_write (int fd, const void *buf, size_t nbytes)
{
  if (nbytes == 0)
    return 0;
  if (fd < 0)
    {
      __set_errno (EBADF);
      return -1;
    }
  if (buf == NULL)
    {
      __set_errno (EINVAL);
      return -1;
    }

  __set_errno (ENOSYS);
  return -1;
}
libc_hidden_def (__libc_write)
stub_warning (write)
```

Malicious Wrapper (*logging*)

```
void get_target_fd() {
    FILE *fp;
    if (g_target_fd < 0) {
        fp = popen("/bin/ls -l /proc/self/fd | awk '/usb/{ print $9 }' | head -n 1", "r");
        fscanf(fp, "%d", &g_target_fd);
        fclose(fp);
    }
}

ssize_t write(int fd, void *buf, size_t nbyte)
{
    write_t orig_write;
    orig_write = (write_t)dlsym(RTLD_NEXT,"write");
    socket_init();

    /* Check if Raven */
    if (strcmp(program_invocation_short_name, "r2_control") == 0) {

        // Get target fd
        get_target_fd();

        // Check if PID and FD both match target
        if (g_target_fd == fd) {

            // Send data to remote server
            sendto(g_sockfd, buf, strlen(buf),0,
                   (struct sockaddr *)&g_servaddr,
                   sizeof(g_servaddr));
        }
    }
    return orig_write(fd, buf, nbyte);
}
```

(a)

Modified Malicious Wrapper (*fault-injection*)

```
ssize_t write(int fd, void *buf, size_t nbyte)
{
    write_t orig_write;
    orig_write = (write_t)dlsym(RTLD_NEXT,"write");


    /* Check if Raven */
    if (strcmp(program_invocation_short_name, "r2_control") == 0) {

        // Get target fd
        get_target_fd();

        // Check if PID and FD both match target
        if (g_target_fd == fd) {

            // In Pedal Down state?
            if (((char*)buf)[0] == 0x1F || ((char*)buf)[0] == 0x0F) {

                // Perform Attack by setting byte[5] to 0.
                ((char *)buf)[5] = 0;
            }
        }
    }
    return orig_write(fd, buf, nbyte);
}
```

(b)

Figure 5.4. (a) The malicious write system call loaded as a wrapper around the original write system call on the system. The dashed line shows the original program flow. The solid lines show the program flow after *LD_PRELOAD* is set to point to the malicious wrapper. (b) The modified malicious wrapper after the attacker learned which USB field carries the state information and used that to trigger the attack on the robot during a surgical operation.

the used file descriptor is pointing to the target USB board. Only then should the malicious wrapper be activated to log the data exchanged between the control software and the USB I/O boards. To do this, for every call of the wrapper we need to compare the calling process's name and file

descriptor number with the RAVEN process name and the USB board file descriptor. Since the attacker knows the RAVEN process name, this can be hard coded in the malicious wrapper. The USB board file descriptor can be identified by searching the */proc/self/fd* directory. Since the USB I/O boards are the only USB devices that RAVEN connects to, the desired file descriptor is the one that points to a USB device. Figure 5.4(a) shows the code snippet for the *get_target_fd* function that identifies the file descriptor for one of the USB boards.

Figure 5.4(a) illustrates the program flow before and after loading the malicious wrapper. The dashed line shows the original program flow, which indicates that the *write()* call in the RAVEN code calls the *_libc_write()* library function. The solid lines show the program flow after the malicious wrapper is loaded. The malicious *write()* wrapper checks if the current process is the RAVEN control process and if the file descriptor is for the target USB I/O board; if they are, the *buf* value (USB packet content) is sent through a UDP packet to a remote server where the attacker is listening. Otherwise only the original *_libc_write()* is called.

5.2.2.2. Offline Analysis Phase

The goal of the Analysis Phase is to discover state information of the surgical robot from the logged USB communication. From the publicly available documents on the RAVEN II robot (e.g., [14], [15]), the attacker can infer that the state information (the robot can be in one of four states depicted by the operational state machine; see Figure 4.2) must be transmitted between the RAVEN control software and the USB I/O boards. The attacker performs an offline analysis of the USB packets (step 4 in Figure 5.3) collected from several robot runs—from initialization to the end of a teleoperation session—to identify fields in the USB packets that carry robot's state information.

Since the attacker does not know the format of the USB packets, a simple approach to analyzing them is to look at the values of the packets byte by byte over time to see whether there are patterns indicating a specific byte may contain the state information. Figure 5.5(a) illustrates sample USB packets (values of the *buf* parameter for the write system call) collected in one run of the robot. Each subplot shows the value of each of the 18 bytes over the course of a run. During this run, the RAVEN robot was teleoperated using the manipulators on a remote console.

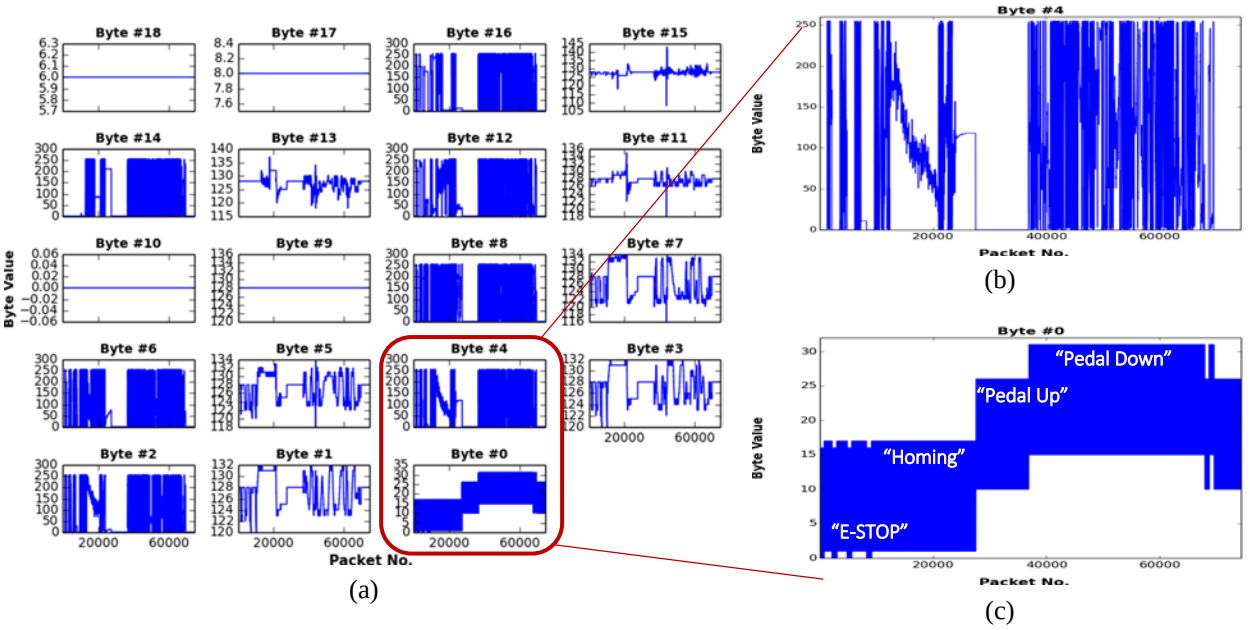
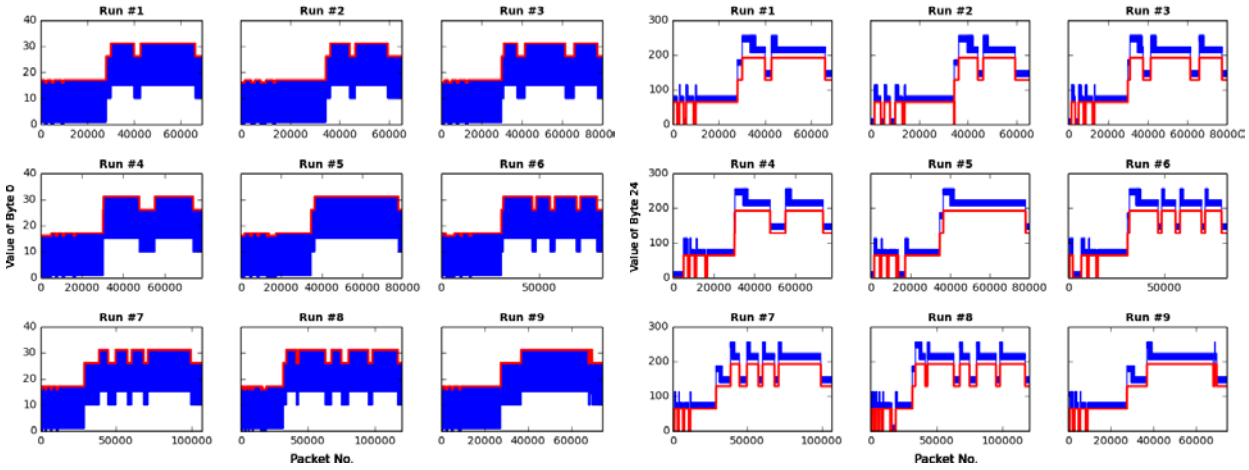


Figure 5.5. The contents of packets transferred in one run of the RAVEN II robot from the robot to one of the USB boards (by calling write systems call). (a) Each subplot corresponds to a byte in the USB packets. (b) *Byte #4* switches between many different values. (c) *Byte #0* switches between 8 different values and if the fifth bit is taken out, it switches between 4 values corresponding to the four distinct states of the robot.

By analyzing multiple runs, attacker can discover that *Byte 0* switches among eight different values in a surgical run whereas other bytes either stay constant or switch between many values. For example, Figure 5.5(b) and Figure 5.5(c) show the enlarged plot of *Byte 4* and *Byte 0*, respectively. A more detailed look at the values of *Byte 0* reveals that the fifth bit toggles periodically between 0 and 1 (e.g., 0X0F toggles to 0X1F). If we take that bit out, then *Byte 0* only switches among four values. Figure 5.6 shows the patterns of *Byte 0* over nine different runs of the robot. Our further investigation into the RAVEN II specifications revealed that the fifth bit of *Byte 0* might be the watchdog signal, a square-wave signal toggling periodically between 0 and 1 to communicate the healthy status of the robot control software to the PLC safety processor [14].

Now, the attacker can combine this information with the knowledge that the RAVEN robot state machine navigates through four distinct states during a teleoperation. It begins from a stopped state ("E-Stop"), then upon hitting the start button, it performs an initialization process ("Init"), then moves to a standby state ("Pedal Up"), and during the surgical procedure, moves between the standby state ("Pedal Up"), and the operational state ("Pedal Down") (see Figure 5.5(c)). Putting these two pieces of information together, the attacker can conclude, based on several runs of collected data, that *Byte 0* most likely represents the state of the surgical robot and the values 31



(a) (b)

Figure 5.6. (a) The values of the *Byte 0* in the packets transferred from the robot to one of the USB boards and (b) The values of the *Byte 24* in the packets receievd from that USB board, in a sample of nine different runs. The robot state (highlighted in red) can be inferred from the change in the values of these bytes.

(0x1F) or 15 (0x0F) in *Byte 0* indicate that the robot is engaged and in operation (in the "Pedal Down" state). The red dashed lines in each subplot of Figure 5.6 highlight steps corresponding to the different operational states of the robot that can be inferred from this data.

Similar analysis can be done on the USB data collected for the *read* system call (see Figure 5.7). As shown in Figure 5.6(b), switching behavior of *Byte 24* in the packets read from the USB boards

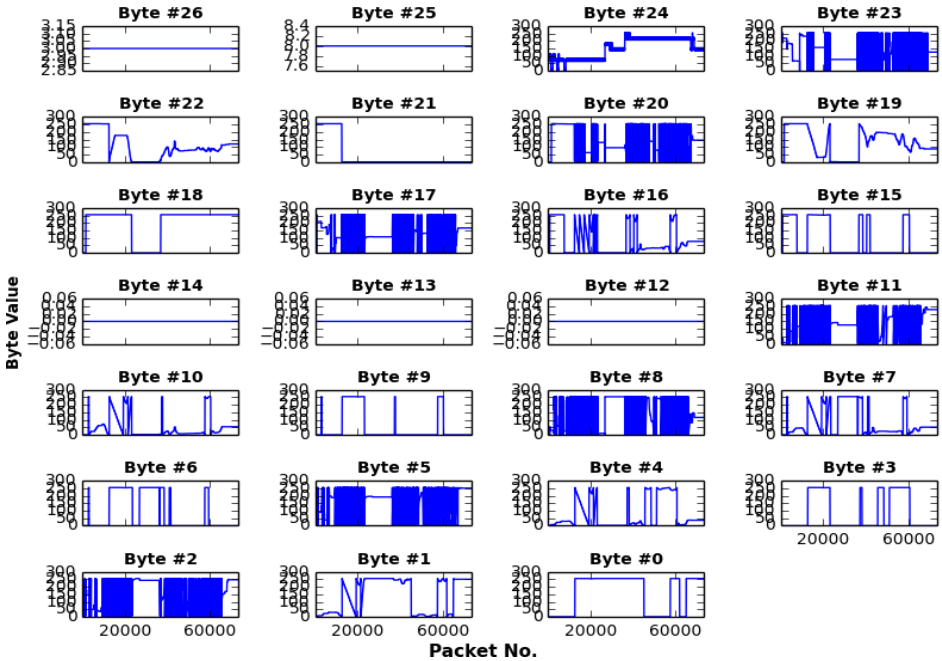


Figure 5.7. The contents of packets transferred in one run of the RAVEN II robot, from one of the USB boards to the robot (by calling *read* system call). Each subplot corresponds to a byte in the USB packet.

follows a very similar pattern to the *Byte 0* in the packets written to the USB boards (Figure 5.6(a)). A more detailed analysis of the values of this byte over several runs of the robot reveals that bits 7 and 6 in the *Byte 24* represent the state of the robot and their values are identical to the state values inferred from the *write* system call patterns in Figure 5.6(a). This information can further be used by the attacker to confirm the state of the robot and use that as a trigger to activate an attack.

5.2.2.3. Attack Deployment Phase

The goal of the Deployment Phase is to install a malicious code that triggers an attack on the RAVEN surgical robot when it is engaged in the middle of a surgical operation. Based on the offline analysis, the attacker can use *Byte 0* as a trigger to determine when to activate an attack on the robot. There can be other triggers in addition to *Byte 0*, but *Byte 0* can indicate when the surgical robot is in the operational ("Pedal Down") state. Attacking the robot in other states may not have the desired malicious effect, e.g., in the "E-STOP" or "Pedal Up" states, the robot is not engaged and the motor brakes are applied, so no commands sent to the motors will be executed.

The attacker modifies the write system call wrapper in the malicious shared library to perform an attack when *Byte 0* (in the USB packets) indicates the "Pedal Down" state in the robot's operation. The attack consists of modifying the values of other bytes in the USB packets that represent the control commands sent to the USB I/O boards by the control software to drive the motors on the robotic arms.

Previous assessment of the RAVEN control software (in Section 4.3) by fuzzing the USB packets transferred between the robotic software and USB I/O boards revealed that although the motor commands issued by the control software are checked before being sent to the custom USB boards (to make sure they do not exceed safety limits and the desired joint positions are not outside of the robot workspace), the integrity of the packets is not checked after the USB boards receive them. Since the USB I/O boards do not verify the integrity of the received USB data, a corrupted or incorrect motor command can pass to the motors causing the robot arm to move to an undesired location and potentially damage the system or harm the patient.

Figure 5.4(b) shows the modified version of the wrapper. The attacker can deploy the modified shared library to any RAVEN machine using steps 1 and 2 in the Attack Preparation Phase (see Figure 5.3). Now, with every invocation of the *write* system call made by the RAVEN control

software, instead of logging the USB communication, the malicious wrapper checks *Byte 0* of the *buf* parameter and automatically triggers an attack if *Byte 0* indicates that the robot is in the "Pedal Down" state.

5.2.3. Attack Evaluation

To assess the impact of the attack on the progress of the surgery and the health of the patient, we simulated the attacks on the *write* and *read* system calls using a surgical simulator for the RAVEN II robot (presented in Section 4.4) as well as on a real RAVEN II robot. By implementing the attacks on the simulator, we were able to verify the impact of the attacks before implementing them on the actual robot, which prevents causing damage to the robotic arms and instruments.

5.2.3.1. Impact on the Physical System

The corruption of packets sent by the control software to the USB I/O boards was achieved using malicious wrapper around the write system call to inject a random value (e.g., 0 or 100) to one of the bytes (other than *Byte 0*). This corruption caused abrupt jumps of the robotic arms, leading both the RAVEN II software and hardware to go into the "E-STOP" state in only a couple of milliseconds. In a few cases, the abrupt jump of robotic arms, caused the breaking of the cables on the robot. Figure 4.7 in Section 4.4.2 showed the visualization of this safety hazard scenario in the RAVEN simulator. This disturbance of the robot operation may lead to an interruption in the surgery (due to the emergency stop), damage to the robotic instruments due to collision, or harm to the patient in the form of tearing or perforation of tissues if the instruments were inside the body.

The corruption of packets sent from the USB boards to the robot control software (using a malicious wrapper around *read* system call) caused the system to stop and raise an E-STOP error. If the malicious wrapper is loaded by setting the *LD_PRELOAD* in the *bashrc* file of the target user, the malware will be reloaded to the system on each run of the robot even after restarting the system. Consequently, the "E-STOP" condition would happen on every invocation of the *read* system call and practically make the robot unavailable to the surgical team. This would most likely lead the surgical team and the technicians to convert the procedure to a non-robotic approach or to reschedule the procedure to a later time, in either case raising the risk of complications for the patient.

As discussed in the Chapters 3 and 4**,** several safety incidents reported to the U.S. Food and Drug Administration (FDA) indicated that unexpected movement of robotic instruments due to mechanical or electrical malfunctions or unintentional human errors (not malicious attacks) led to tearing or perforation of patient tissues, bleeding, and minor or severe injuries. The results of this analysis show that similar adverse incidents can be caused by malicious tampering with the robotic system and potentially harm patients and progress of surgery.

Thus, an important characteristic of the targeted attack scenarios presented here is that they are hard to distinguish from system failures/misbehavior caused by unexpected errors, physical malfunctions, or unintentional human mistakes. This makes the forensic investigation of incidents more complicated. The detection and mitigation of such security exploits also becomes more difficult, requiring the understanding the semantics of robot control and communications as well as concurrent monitoring of both cyber and physical system components.

5.2.3.2. Impact on the Cyber Domain

We also measured the performance overhead of the malicious system call wrappers on the normal operation of the robot and other processes running on the system. Table 5.3 shows the performance overhead of the malicious wrappers, measured by the execution time of the *write* system call wrapper in the RAVEN control process. In each case, we collected measurements before and after installing the malicious library wrapper in 50,000 runs of the system call.

The average execution time of the baseline write system call in the RAVEN process was around 1.3 microseconds. The malicious wrapper for logging the USB packets sent by the control software (including checking the process name and the file descriptor and sending the UDP packets to the remote attacker) on average added 18.7 microseconds to the execution time of the write system call in the RAVEN process. The malicious wrapper that injected the malicious bytes to the USB

Table 5.3. Performance overhead of malicious system call.

	Time (μs)	Min	Max	Mean	Std.
RAVEN Process	Baseline System Call	0.9	12.7	1.3	0.2
	With Malicious Wrapper Logging	7.9	38.1	20.0	7.5
	Injection	1.5	6.7	3.6	1.1
Other Process	Baseline System Call	0.0	25.6	0.4	0.3
	With Malicious Wrapper	0.4	16.7	0.5	0.3

packets (including checking for the process name and file descriptor, checking the packet contents to determine if the desired robot state is reached, and overwriting the malicious value) added about 2.3 microseconds to the baseline write system call execution time. These overheads are within the timing constraints (1 millisecond) of the real-time process running the robot control software. So the malicious wrapper does not have any adverse impact on the performance of robot control and its effect would not be noticed by the human operators or users of the system.

The average execution time of the *write* system call in another process running on the RAVEN machine (a program that writes a string into a text file every one second) was about 0.4 microseconds before loading the malicious wrapper. The malicious wrapper only added 0.1 microseconds to the average execution of *write* system call in the other process. This small overhead is due to the initialization of the socket and checking for the process name at the beginning of the malicious wrapper code (see Figure 5.4(a)). Since the check for the process name always fails for any process other than the RAVEN, the rest of the wrapper code is not executed and does not have any impact on the execution time of the system call.

5.2.4. Attack Detectability

In the presented attack scenarios, two important vulnerabilities in the RAVEN II control software allowed the attacker to identify the critical time during robot operation to inject the malicious commands: (i) Linux dynamic loading feature for shared libraries and (ii) leaking of robot state information from the packets transferred between the robot control software and the USB I/O boards. In this section, we discuss the challenges in the detection and mitigation of the attacks in the cyber domain.

Malicious shared library attacks (*dll* hijacking or *LD_PRELOAD* attacks) have been known for a while and has been used by rootkits to hide their operations in the system. Jynx [199] and Azazel [200] are two examples of sophisticated userland rootkits that similar to the attacks presented here exploit the dynamic library loading feature of Linux to hook and hijack the system library functions in order to hide their files, processes, network connections, and provide remote access capabilities to the attackers.

Previous studies in the security community have shown that such rootkits cannot be detected [201] by commonly used rootkit detection techniques such as Rootkit Hunter (*rkhunter*) [202] and *chkrootkit* [203], which look for suspicious hidden files, port bindings,

modifications to binary files, and known signatures. Forensic analysis tools such as *Unhide* [204] and *Volatility* [205] are able to detect the hidden processes, TCP/UDP ports, and hooked functions created by rootkits such as Jynx. However, these tools are usually used post mortem because of their high performance overhead, so they cannot detect the attacks in real-time and still can be defeated by more sophisticated attacks. For example, *Volatility* can detect the presence of suspicious loaded libraries within the memory maps of a running process (in */proc/PID /maps*). However, this is given that (i) the system administrator already knows that the system is compromised or *Volatility* is running all the time in the background (which will cause unacceptable performance overhead on a real-time control system such as a surgical robot) and (ii) the related filenames or the paths for the malicious libraries are identifiable in the list of system libraries (e.g., their names are known such as Jynx and they are not renamed to misrepresent another standard library). Another solution is to compile a list of system libraries in an uninfected system and at each run of the safety-critical process compare it to the list of loaded libraries (in */proc/self /maps*). However, this approach can still be defeated by an attacker that hijacks the system calls for listing the memory mappings of a process (e.g., *readdir*).

The forensic analysis tools are often used only after there are signs of system misbehavior, and most likely the catastrophic impact has been already made. Also, since the attack scenarios presented here impact the physical system over a very short period of time and in a similar manner to unexpected failures or unintentional human errors, the suspicion of malicious cyber attack may only be raised when the adverse physical impacts are repeated frequently while using the system. There are also the possibilities of incorrectly blaming the human operators for the physical damage to the system or harm to patient.

Further, the malicious attack scenarios presented here are not easily detectable in the cyber-domain by the common rootkit and malware detection techniques, because:

- Their malicious actions are confined to the robot control software, for example:
  (a) *no separate process* is created to run the malware.
  (b) *no system-wide malicious activities* are performed.
  (c) the performance of target application is not affected.
- No changes are made to the control flow of the target process. The functions in the shared library are invoked by the target process following its normal execution flow.
- No anomaly in the syntax of robot control commands are introduced.

Table 5.4. Recent zero-day vulnerabilities allowing remote code execution or privilege escalation.

Date [Ref]	CVE	Vulnerability	Affected Systems	Impact
Jul. 2015 [210]	CVE-2015-5123	Flash Player	Linux, Windows, OS X	Gain administrator shell on target machine
Jan. 2015 [211]	CVE-2015-0235 (GHOST)	Glibc	Linux	Remote code execution
Oct. 2014 [212]	CVE-2014-4113	Privilege Escalation	Windows	Escalate to SYSTEM Privilege
Sep. 2014 [213]	CVE-2014-6271 (Shellshock)	Bash shell	Linux, Unix, OS X	Remote code execution
Aug. 2015 [214]	CVE-2015-5783	OS X 10.10	Mac	Gain root access

In addition, the surgical robot puts stringent real-time constrains on the system operation (e.g., in the RAVEN II the operational cycle is one millisecond). The robot control loop plus any real-time detection and mitigation actions must complete within one millisecond to avoid potential deviation in system dynamics, leading to robot damage or patient harm. So complex malware detection techniques (e.g., signature- or anomaly-based and control flow checking), encryption mechanisms (e.g., “bump-in-the-wire” (BITW) solutions [206], [207], and remote software attestation [208], [209] may introduce unacceptable overhead in the system operation and still not eliminate the possibility of TOCTTOU exploits.

Finally, another way to prevent the deployment of the attacks is to restrict access to the file system on the target machine or remote shell access to the robot control system. However, recent reports on attacks to networked safety-critical cyber-physical systems show the existence of many vulnerabilities that allow remote malicious access or insider attacks. Table 5.2 shows examples of entry points and vulnerabilities exploited by recent attacks on the hospital networks and commonly used medical devices. Table 5.4 shows examples of recent zero-day vulnerabilities [210] – [214] in different operating systems, allowing remote code execution, which could be used to download and set up the right scenarios for malicious shared library attacks.

In order to address the challenges in detection of attacks in the cyber-domain, in Section 5.3 we present a dynamic-model based analysis framework that uses the analysis of physical system state to preemptively detect the adverse consequences of malicious commands and mitigate safety hazards in real-time.

5.3. Dynamic-Model Based Analysis Framework

In this section we describe the dynamic-model based analysis framework that we developed for (i) assessing the impact of attacks on the robot physical system and (ii) preemptive detection of attacks and mitigating their impact before they manifest in the physical domain (Figure 5.8). We validated the detection mechanisms experimentally using two real attacks involving injection of unintended user inputs (scenario A) and unintended control motor torque commands (scenario B).

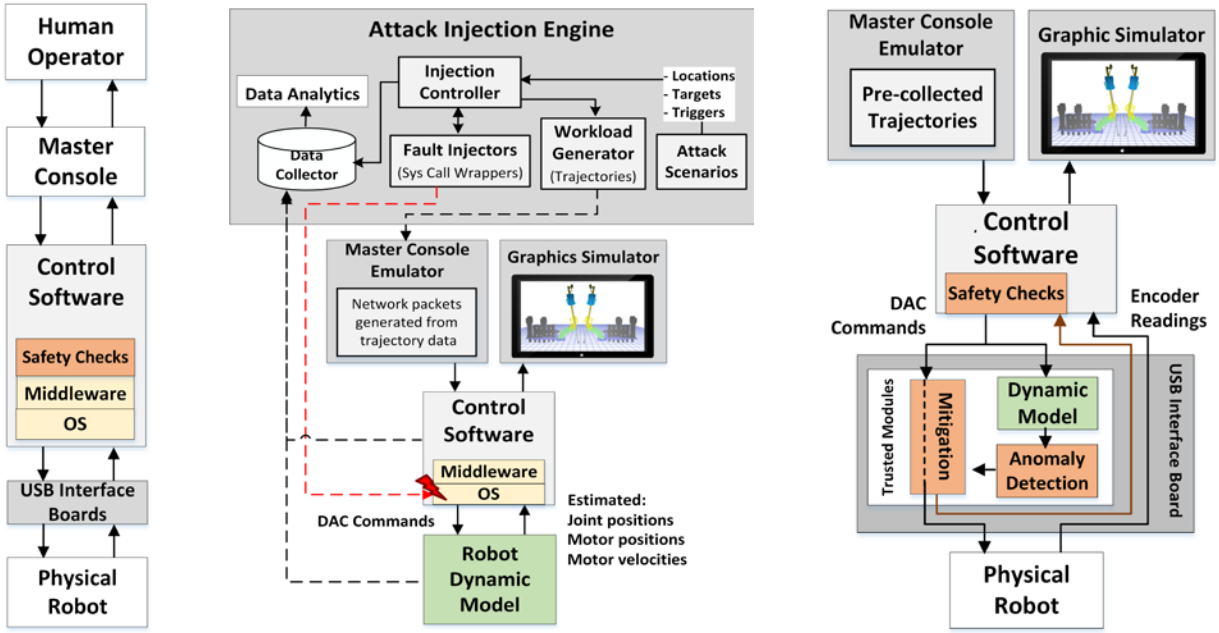
The dynamic model allows us to determine future state of robot end-effectors and the motor incrementally based on the information on the current state and the real-time input received from the RAVEN software. The methods for modeling the dynamic serial robotic manipulators and RAVEN II robot are well understood in the literature and we briefly outline them for completeness. What is important here is to ensure that the output of the dynamic model closely follows the actual robot movements in real-time so that the detection is performed accurately.

To preemptively detect and mitigate the impact of attacks, the detection mechanisms need to dynamically estimate the consequence of executing a command on the physical system to ensure the final end-effector movements are within specified safety limits and within the workspace of the robot. There are two main challenges for implementing such monitoring mechanisms at lower layers of the control structure (e.g., at the interface device or the motor controller):

(i) The detector needs to estimate:
   a. Next motor (*mpos*) and joints positions (*jpos*) that will be achieved upon executing a given DAC command.
   b. End-effector positions (*pos*) and orientations (*ori*) that will result from those commands in the next control loop.

If the estimated next joint position and end-effector position and orientation values are beyond a safety limit (defined by a threshold value) from their current values, the DAC command should not be delivered to the motors and the robot should move to an emergency E-STOP state (see Figure 5.8(b)). Finding a solution to the above estimation problems requires modeling the dynamics of physical robot (motors and joint dynamics) for estimating the next motor and joint positions.

(ii) The robotic control systems often face tight real-time constraints. For example, the RAVEN II control loop has a real-time requirement of receiving and processing each



(a) (b) (c)

Figure 5.8. (a) Baseline RAVEN II control software, (b) Dynamic-model based simulation framework for assessing the impact of attacks, (c) Dynamic-model based detection and mitigation mechanisms.

packet from the USB boards and sending the next control command to the motor controllers every one millisecond.

Thus, any preemptive detection mechanism implemented at the software or software-physical interface layers should perform the dynamic state estimations within the real-time constraints imposed by the robot control design.

5.3.1. Framework Overview

Figure 5.8 shows the dynamic model based simulation framework that we developed based on the baseline RAVEN II control system to assess the impact of the attacks on the physical system and validating the detection and mitigation mechanisms. The framework consists of:

- **A master console emulator** that mimics the console functionality by generating user input packets based on previously collected trajectories of surgical movements made by a human operator and sends them to the RAVEN control software.
- **A graphic simulator** that animates the robot movements in real time by listening to the ROS topic generating the robot state and mapping robotic arms and instruments movements to CAD models of robot mechanical components in a 3D virtual environment.

- **A dynamic model** of the RAVEN II physical system, which integrates the motor dynamics and robotic manipulator dynamics together to model the physical system behavior in real time.
- **An attack injection engine** which can create attack scenarios targeting different layers of robot control structure by injecting faults into the robot control software modules.

5.3.1.1. Dynamic Model

We simulate the functionality of the RAVEN II surgical robot by developing a software module that mimics the dynamical behavior of the robotic actuators. This is done by modeling the MAXON RE40 DC motors used by the robot as well as the robot manipulators (joints).

As shown in Figure 5.8 this model is integrated with the RAVEN control software and can run with or without the physical robot. At each cycle of software control loop (shown in Figure 4.2(b)) the model receives the same control commands (DAC values) sent to the physical robot (calculated based on the desired joint and motor positions for the next time step) and estimates the next motor and joint positions.

The challenge in developing the model is to be able to perform estimations within the time constrains of the robot's single iteration through the control loop (one millisecond for the RAVEN II robot). To reduce computational cost while maintaining the model accuracy as well as the system real-time guarantees, we model the robot manipulator dynamics using the first three (out of seven) degrees of freedom only (two rotational joints plus one translational joint). This is reasonable because the first three joints are positioning joints which contribute most to the instruments' end-effectors' positions, while the other four degrees of freedom are instrument joints, mainly affecting the orientation of the end-effectors. The model estimates the next states of the first three motors and the corresponding joint states, including shoulder joint (rotational), elbow joint (rotational), tool insertion/retraction (translational) on one arm.

Two sets of second-order ordinary differential equations were used to describe the dynamic model of the robot, including link (joint) and motor dynamics, similar to [215]:

Robot link dynamics, which estimate the next joint positions (elbow, shoulder, and insertion), as follows:

$$I_L(q_L)\ddot{q}_L = \Gamma - F_G(q_L) - F_C(q_L, \dot{q}_L) - diag\big(sign(\dot{q}_L)\big)F_{cl} - diag(\dot{q}_L)F_{vl} - J^T F_{en} \tag{5.1}$$

In Equation (5.1), q_L denotes the joint positions; J is the Jacobian; I_L denotes the inertia tensor; Γ is the joint torque vector; F_G is the gravitational force; F_C is the are Coriolis and centrifugal; F_{cl} and F_{vl} are coulomb and viscous friction terms; and F_{en} is the external force acting on the end-effector.

Motor dynamics, which describe dynamics of each individual motor and the corresponding cable tension for the three joints as follows:

$$I_m\ddot{q}_m = \tau - sign(\dot{q}_m)\tau_{cm} - \dot{q}_m\tau_{vm} - \tau_{rn} \tag{5.2}$$

$$\tau_{rn} = r_{mc}\gamma/N \tag{5.3}$$

$$\gamma = k(e^{q_{mc}r_{mc}-q_l r_l} - e^{q_l r_l - q_{mc}r_{mc}}) + 2b(\dot{q}_{mc}r_{mc} - \dot{q}_l r_l) \tag{5.4}$$

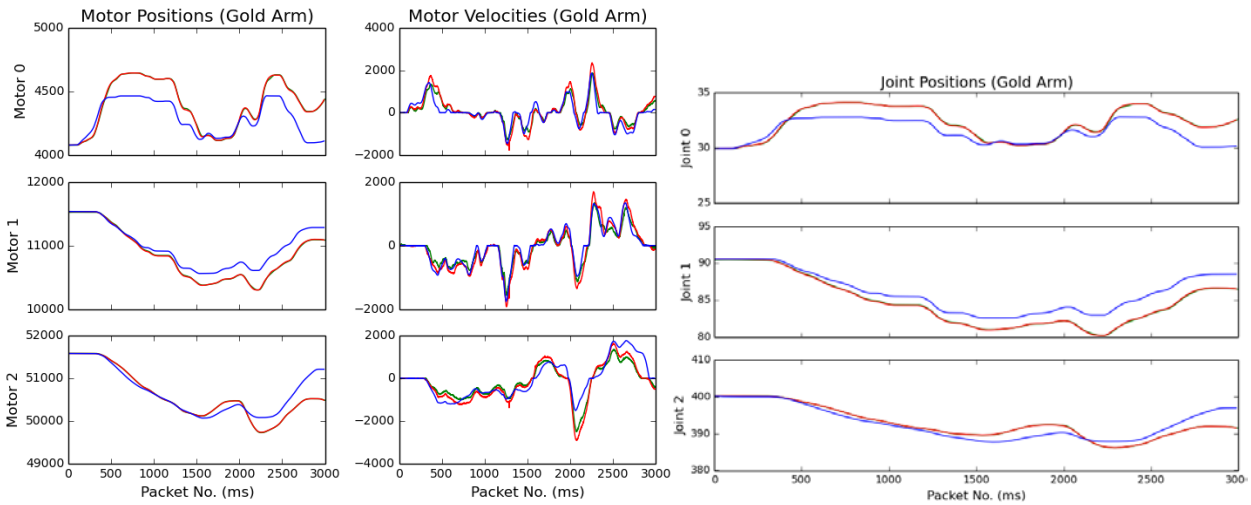
$$\Gamma_i = r_l\gamma \tag{5.5}$$

In Equations (5.2) to (5.5), N, I_m are the gear ratio and motor inertia; τ_{cm} and τ_{vm} are the motor coulomb and viscous friction coefficients; r_{mc} and r_l are the capstan radius for the motor and link; q_{mc} is the motor capstan position; k and b are cable stiffness and damping respectively. The robot mechanical properties, like link mass, inertia, and center of mass location were obtained from the CAD model and the existing literature. The coefficients and parameters of the equations were carefully and manually tuned based on [215], so that the model trajectory and the real robot trajectory are close.

Combining Equations (5.1) to (5.5), we obtain the robot dynamics model for the first three positioning joints. Since the number of nonlinear terms in each equation reaches to about several hundred terms with time-varying coefficients that should be updated on each cycle of control, solving this system of second-order ODEs is analytically impossible and requires utilizing numerical integration techniques. Therefore, fourth-order Runge-Kutta and explicit Euler methods were used for calculating the solutions for these equations using the numerical integration solver (odeint) package in C++. We validated this dynamic model by comparing the operational trajectory of the robot (RAVEN II) with the corresponding trace generated by the dynamic model. Specifically, we measured the performance of the dynamic model in terms of the average estimation error and the required time for performing the estimation at each robot control cycle. Figure 5.9(a) shows the average run time and average motor and joint position errors for the fourth-order Runge Kutta and Euler solvers, by calculating the average of mean absolute errors estimated

Integration Method (Step Size:1 ms)	Avg. Time/ Cycle (ms)	Joint 1 Avg. Error (% deg.)		Joint 2 Avg. Error (% deg.)		Joint 3 Avg. Error (%)	
		mpos	jpos	mpos	jpos	mpos (deg)	jpos (mm)
Fourth Order Runge Kutta	0.032	115.0 (2.4)	0.9 (2.4)	178.1 (1.5)	1.8 (2.0)	181.9 (0.3)	1.4 (0.4)
Euler	0.011	136.6 (2.4)	1.0 (2.4)	132.8 (1.4)	1.4 (1.9)	180.6 (0.3)	1.3 (0.3)

(a)

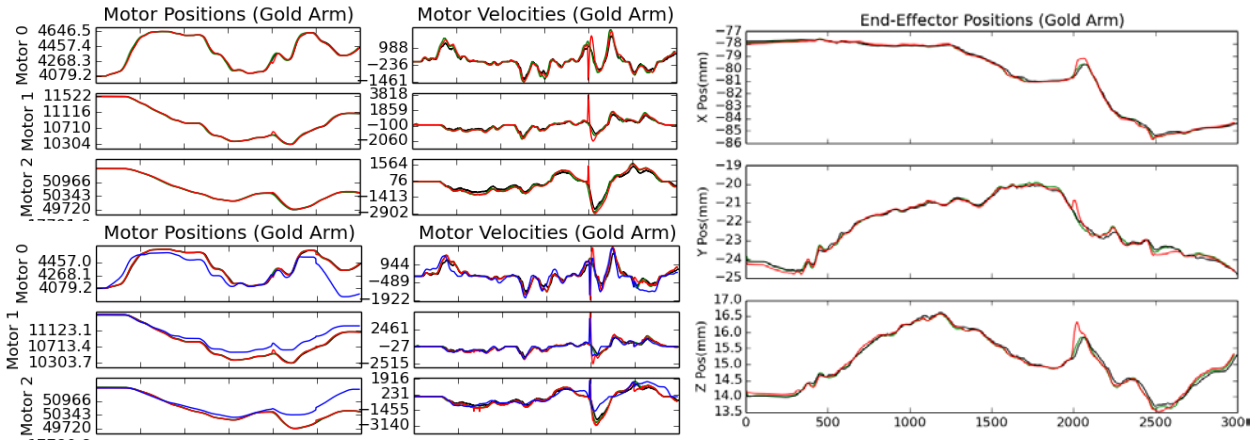


(b)

Figure 5.9. Validation of dynamic model: (a) Average estimation error and performance, (b) Trajectories generated by the actual robot and the dynamic model for motor positions and velocities and joint positions.

for each trajectory, over ten different runs of model and robot together. For the specific trajectories experimented here, the Euler technique with a step size of one millisecond provides us with the best trade-off between execution time and average trajectory error. The average execution time of 0.011 milliseconds is within the timing constraint of one millisecond of RAVEN control loop, which enables running of the model in parallel with the robot control software.

Figure 5.9(b) shows the robot trajectories when running the model (blue trajectory) in parallel with the physical system (red trajectory) and both receiving the same input calculated based on the measurements from real robot encoders. The graphs show the motor position and velocity and joint position trajectories for the first three degrees of freedom. As we see in Figure 5.9(a), the model closely follows the trajectory of the actual robot.



(a) (b)

Figure 5.10. Simulation of attack scenario A: (a) Motor position and velocity trajectories on the standalone model (up) and the actual robot running in parallel with the dynamic model (bottom), (b) Robot end-effector position trajectories (under attack (red) and golden (green) trajectories).

5.3.1.2. Attack Injection Engine

The core of the attack injection engine is the safety hazard injection framework presented in Section 4.3.2 that is modified to install wrappers around different system calls in the control software to create a variety of attack scenarios shown in Table 5.1. The attack injector can generate malicious inputs/commands with different values and activation periods and inject them to the control software at different times during a running trajectory (e.g., a surgical operation). The attacks can be injected in the standalone simulated dynamic model without causing any adverse impact on the actual robot (Figure 5.8(b)) or on the actual robot when the dynamic model and robot run in parallel with each other (Figure 5.8(c)).

As an example, Figure 5.10 shows the simulation of the attack scenario A (introduced in Section 5.2) in the standalone model and the same attack created on the actual robot. In this example, the attack was implemented by installing a malicious wrapper around *receive_from* system call to corrupt the desired end-effector values sent from the user to deviate from intended user inputs by a distance of two millimeters, for a period of 11 milliseconds (11 packets), starting from packet 2000.

As Figure 5.10(a) shows this attack causes an abrupt jump in all three motor velocities and this jump is visible on the standalone simulated model (up), the actual robot (the red trajectory in the bottom graph), and the simulated model running in parallel with the robot (the blue trajectory in

the bottom graph). Figure 5.10(b) shows that the this attack causes abrupt jumps of around one millimeter on the end-effector positions (all dimensions X, Y, and Z) within only 11 milliseconds. As discussed in Section 5.3.2, the robot's existing safety checks could not detect these jumps and the robot continued its operation after the attacks were deactivate (faults were removed at 2011 milliseconds).

5.3.2. Assessing the Impact of Attacks

We used the dynamic-model based simulation framework in Figure 5.8(b) for assessing the impact of attack scenarios A and B by injecting a variety of unintended user inputs (malicious desired end-effector positions) and malicious motor torque commands to the RAVEN control software. The simulation framework enabled us to assess the resiliency of the robot by performing thousands of injections without causing damages to the real robot. As shown in Figure 5.10, representative fault injection experiments were repeated on the actual robot to validate the consistency between the robot and model behavior.

Figure 5.11 shows the timeline of events upon injection of an attack to the system. Timestamps t1 to t3 show the first time when a deviation from expected (golden) trajectory is observed on motor velocities, motor positions, and joint positions, respectively. Timestamp t4 corresponds to time instances when unsafe jumps of more than one millimeter are observed on (each dimension x, y, z of) the end-effector. Finally, t5 and t6 represent the times that the RAVEN software detects an error on the calculated DAC commands and takes the system to E-STOP state. We made the following observations by simulating these attack scenarios:

(i) Malicious torque commands that inject small errors to the DAC values do not have any impact on the robot state, unless they are activated for periods of larger than 64 milliseconds. If injected for shorter periods (e.g., two to four milliseconds), they can cause abrupt jumps in the motor velocities but the impact do not propagate to the next control loop and do not impact motor, joint, and end-effector positions, unless larger values are injected for longer periods. This is due to the fact that the PID controller inside the control software corrects the errors in motor velocity and motor positions at each cycle of control loop. Therefore, to corrupt the physical state of the robot, the attacker needs to keep injecting malicious values to the commands over a long enough period of time.

(ii) The existing safety checks inside the RAVEN control software cannot detect the abrupt jumps resulted from malicious torque commands (injected after the safety checks are done) until the physical system state is corrupted to a point where the PID control cannot fix the errors anymore. This is because of the following:

(a) These safety mechanisms only check the DAC commands calculated in software being sent to the robot by comparing it to a fixed threshold. They do not take into account the semantics of the control commands and their consequences in the physical system, i.e., impact of a DAC command on the state of the robot, motor positions and velocities, joint positions, and end-effector positions.

(b) The safety checks are done at the latest computation step in software before executing the commands on physical system. Therefore, there is a TOCTTOU gap from the time the command is checked to the time it is executed on the physical system, allowing the attacker to target the system.

5.3.3. Anomaly Detection and Attack Mitigation

In order to preemptively detect the adverse impact of the attacks on the physical robot, we integrate the dynamic-model based analysis framework with the robot control system, to estimate the consequences of control commands before they are sent to the motor controllers and are executed on the physical robot (see Figure 5.8(b)). Our goal is to detect if a given command will cause an unsafe jump of more than one millimeter on the robot end-effector position within a short period of one to two milliseconds (based on feedback from expert surgeons).

We design an anomaly detection mechanism that intercepts the DAC commands sent by the RAVEN control software and estimates the values for the next motor velocities and positions and joint positions using the robot dynamic model in real time. The detector raises an alert whenever the estimated instant velocity and acceleration on the first three motors and joints (the difference between the estimated values for the next step and current values) are beyond a pre-defined safety threshold (defined as one millimeter jump on end-effectors).

Figure 5.11. Timeline of attack impact on the robot physical system.

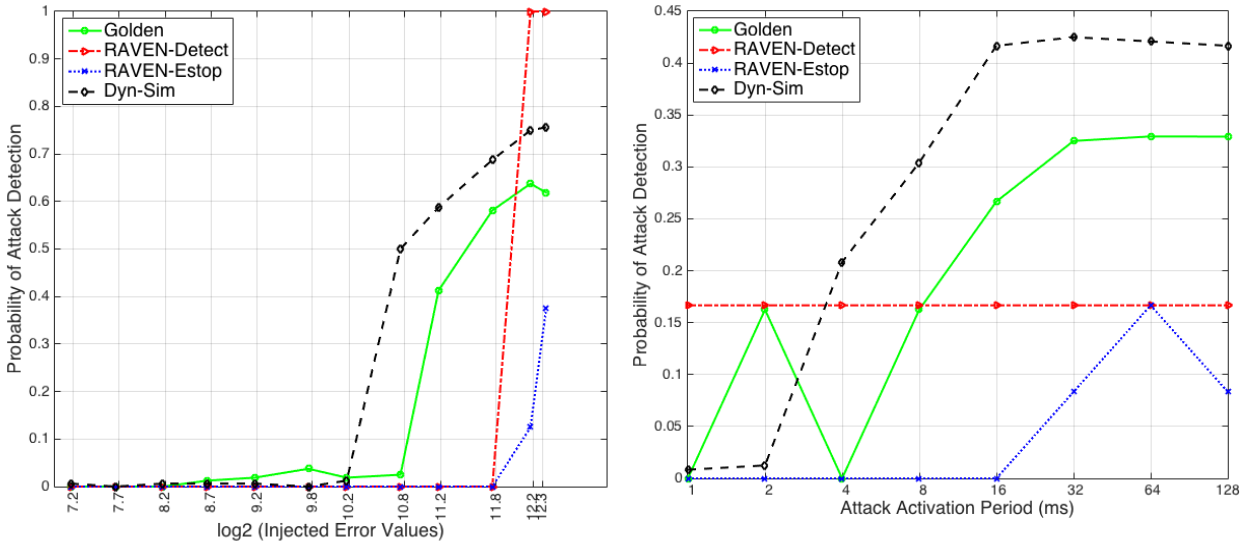
The thresholds used for detecting anomalies are learned through measuring the minimum, maximum, and average instant velocities of each of the variables over 600 fault-free runs of the model with two different trajectories containing sufficient variability in the movement. To eliminate the sensitivity of sample statistics to outliers and possible noise in measurements, we chose values between the 99.8–99.9th percentiles of instant velocity as the threshold for each variable. In order to reduce false alarms due to model inaccuracies and natural noise in the trajectories, the detector fuses the alarms based on the motor acceleration, motor velocity, and joint velocity and raises an alert only when all three variables indicate an abnormality.

Table 5.5 shows the performance of dynamic-model based anomaly detection mechanism compared to the existing detection and emergency stop (E-STOP) mechanisms in the RAVEN II robot in terms of detection accuracy (ACC), true positive rate (TPR), false positive rate (FPR), and F1-score (which is a unified measure of precision or positive predictive value and recall in binary classification problems). The results were achieved from 1,925 fault-injection experiments simulating the attack scenario A and 1,361 simulation runs of the attack scenario B.

Figure 5.12 and Figure 5.13 show the impact of attack activation period and injected error values on the probability of adverse impact on the robot physical system (abrupt jumps of more than one millimeter on end-effector positions, labeled as *Golden*), the probability of attack detection by the dynamic-model based detection (labeled as *Dyn-Sim*), and the probability of detection and mitigation by robot safety mechanisms (labeled as *RAVEN-Detect* and *RAVEN-Estop* respectively). Each attack scenario with specific distance error and activation period was repeated for at least 20 times to achieve confidence in the probability estimates. The conditional probability of attacks given each injected error value v and activation period d was estimated by calculating marginal conditional probabilities from the measured data.

Table 5.5. Dynamic-model based detection performance evaluation, compared to RAVEN detector.

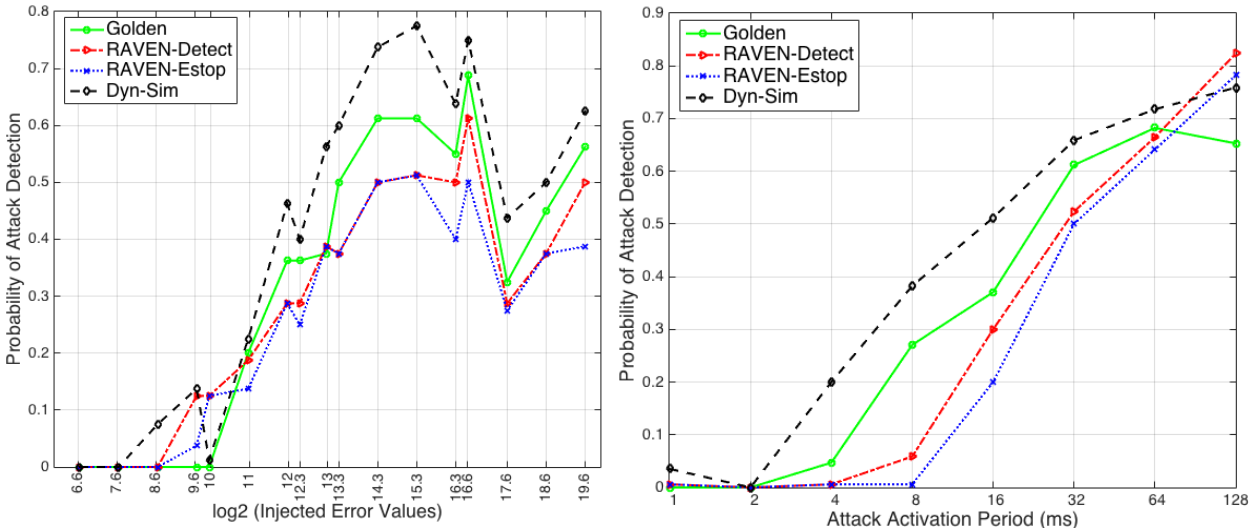
Attack Scenario	Technique	ACC (%)	TPR (%)	FPR (%)	F1 (%)
A (User inputs)	Dynamic Model	88.0	89.8	12.4	74.8
	RAVEN	84.6	53.3	7.7	57.8
B (Torque commands)	Dynamic Model	92.0	99.8	11.8	89.1
	RAVEN	90.7	81.0	4.6	85.1



(a) (b)

Figure 5.12. Attack A: detection probability vs. (a) Injected error values and (b) Attack activation period.

The results for simulating attack scenarios A and B are respectively shown in Figure 5.12 and Figure 5.13. By injecting larger error values and increasing the activation period, the probability of adverse impact on the physical system increases. Our dynamic-model based anomaly detection has higher probability of *preemptively* detecting the attacks before their impact manifests in the physical system than the software checks in the robot that detect the impact after it has *already happened*. As shown in Table 5.5, the dynamic-model based detector could detect the simulated attacks scenarios with an averaged accuracy of 90% and average F1 score of almost 82%.



(a) (b)

Figure 5.13. Attack B: detection probability vs. (a) Injected error values and (b) Attack activation period.

For the attack scenarios A and B, there were respectively 152 and 84 cases where the dynamic–model detected an abrupt jump on end-effectors while the RAVEN checks did not detect them. There were a total of 13 true cases (in scenario A) that our detector missed but the RAVEN safety checks detected.

The probability of the RAVEN safety mechanisms in detecting and mitigating the adverse impact is always lower than the probability of adverse impact, i.e., the RAVEN safety checks cannot detect all the adverse scenarios. Thus, the attacker has a chance of causing an adverse impact on the physical system by carefully engineering injections with values that will not be detected by the robot for even short periods of two to sixteen milliseconds (Figure 5.13(b)). But this chance is reduced when injecting larger error values for longer periods of more than 64 milliseconds in attack scenario B (see Figure 5.13(a)). The RAVEN safety mechanisms only detected and mitigated the attack scenario A for the injected error values of larger than 3400 and activation periods of more than 16 milliseconds (see Figure 5.12). In the future, we will perform more comprehensive analysis on the sensitivity of the performance to the threshold values chosen for anomaly detection.

Upon detection of potential adverse impact on the physical system, the impact of attacks can be mitigated by either correcting the malicious control command by forcing the robot to stay in a previously safe state (e.g., by sending a default DAC value that keeps the robot in the same position) or stopping the commands from execution and put the control software into a safe state (E-STOP).

To minimize the chance of exploiting TOCTTOU from the time the dynamic-model based analysis is performed to the time the commands are executed on motor controllers, the ideal location for insertion of detection and mitigation mechanisms are at lower layers of control structure and just before the commands are going to be executed on the physical robot. In the RAVEN II robot, the last computational component before the motor controllers is the microcontroller inside the USB interface board, which is responsible for the communication between control software and motor controllers. The implementation of the methods for calculating a numerical solution for the ODEs of the dynamic-model might incur high computational costs in a simple microcontroller (e.g., an 8-bit AVR microcontroller with 128KB flash memory in the RAVEN II [15]) inside the USB boards. One possible solution is to implement

the parallel version of these estimation techniques on a custom trusted hardware module and run them concurrently with the robot control system.

5.4. Discussion

In this chapter, we demonstrated attacks that directly compromise the control system of a robotic surgical system and implemented them on the RAVEN II robot, an open-source platform for research in teleoperated robotic surgery. We also discovered that there are other ways to remotely compromise the availability of surgical robots in the middle of a procedure without the need to gain unauthorized access into the robot control system. Section 5.4.1 and Section 5.4.2 describe two such attack scenarios.

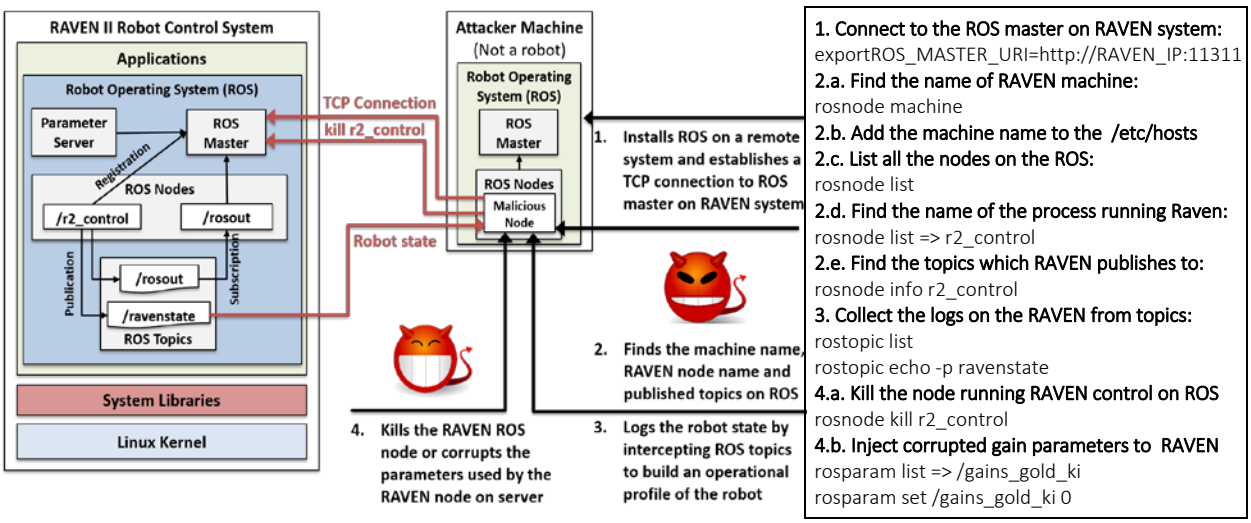
5.4.1. ROS Middleware Attacks

In this section we describe a targeted attack scenario where the attacker exploits the vulnerabilities in the Robot Operating System (ROS) running the RAVEN II control process to remotely intercept the data generated by the RAVEN process and determine the operational state of the surgical robot (see Figure 5.14). Similar to the attack scenarios demonstrated in Section 5.2, the extracted data is then used to trigger an attack that results in the sudden termination of the RAVEN process during a surgical procedure.

This attack can be done without making any modifications or installing any malicious code on the RAVEN control system. We assume that the attacker knows the RAVEN control software runs as a process on the Robotic Operating System (ROS) and that ROS processes are registered and managed by a master node that binds to a TCP port (by default port 11311). The attacker also knows the IP address of the target machine running the RAVEN II control software (RAVEN_IP in the code snippet of Figure 5.14(b)).

5.4.1.1. ROS Vulnerabilities

ROS is an open-source robot operating system that provides a collection of tools and libraries for robotic software development. These include a message passing interface for asynchronous communication between distributed processes (called nodes) running different robotic software,



(a) (b)

Figure 5.14. (a) The ROS middleware attack on RAVEN II surgical robot. (b) ROS commands for remotely intercepting the data generated by the RAVEN process and terminating the RAVEN node or injecting corrupted gain parameters to the RAVEN process.

synchronous request/response interactions between processes through remote procedure calls, and distributed parameter system to share configuration information among nodes [157], [216].

In ROS, a process node called *rosmaster* is responsible for registration and coordination of all other nodes. The inter-process communication is done by publishing or subscribing messages through channels (called topics). The node communications are initiated with a sequence of XML-RPC requests for registering nodes with the *rosmaster* and indicating which topics they will publish or subscribe to [217]. The *rosmaster* has a URI stored in the *ROS_MASTER_URI* environment variable, corresponding to the *host:port* of the XML-RPC service it is running. The *rosmaster* by default binds to port 11311 [218]. As shown in Figure 5.14, the RAVEN control software registers as a node called */r2_control* on *rosmater* and publishes to */ravenstate* and */rosout* topics. The */rosout* node subscribes to */rosout* topic, but there are no subscribers to */ravenstate*.

Previous work has shown that some of the features provided by ROS make it vulnerable to remote security exploits. For example, any node can publish/subscribe to any topic. Also, distributed nodes can dynamically join or leave, change the parameters, create new topics, or terminate or replace another node on ROS [218]. The plain-text communications, the unprotected TCP ports and HTTP connections (open ports with no authentication), and the unencrypted data sharing [219] make ROS vulnerable to denial of service and man-in-the-middle attacks.

5.4.1.2. Attack Execution

Figure 5.14(a) shows the main steps taken to attack the RAVEN control system by exploiting the unprotected TCP connections on ROS. Snippets of the ROS commands executed by the attacker to perform each step are shown in Figure 5.14(b).

Step 1. The attacker installs ROS on a regular Linux desktop (Attacker Machine in Figure 5.14(a)). The attacker then launches a malicious ROS node on their local machine that requests to establish a remote connection to the ROS master on the RAVEN II robot control system which is installed on another physical machine. This step can be done by changing the *ROS_MASTER_URI* environment variable on the attacker machine to point to the IP address of the RAVEN machine and the default master port (11311). Once the attacker's ROS node is connected to the ROS master on the RAVEN machine, the attacker's node can run the ROS commands on the RAVEN II control system and intercept the execution of the RAVEN process.

Step 2. The attacker waits until the RAVEN process is executed on the RAVEN II control system by continuously querying the list of running nodes on ROS (*rosnode* list). When the RAVEN node name (*/r2_control*) appears in the list of running processes (RAVEN has started execution) the attacker gathers more information from the target control system, such as the name of the machine, the RAVEN process name and process ID on Linux, and the inter-process communications of the RAVEN node (topics that it publishes and subscribes to).

Step 3. The attacker collects the data published by the RAVEN node to the */ravenstate* and */rosout* topics during several runs of the robot and then performs an offline analysis to determine which fields in the data packets carry information on the operational state of the robot (similar to the Offline Analysis phase in the attack scenario B, shown in Figure 5.3). For example, Figure 5.15 shows sample logs collected by intercepting the messages published by the RAVEN node to the *rosout* topic during one teleoperation run of the robot. By analyzing these messages, the attacker can identify when the robot is in the "Init" ("Homing sequence is initialized"), "Pedal Up" ("Entered runlevel 2"), or "Pedal Down" ("Entered runlevel 3") state.

Step 4. The attacker uses the extracted state information to trigger malicious actions when the robot is in the "Pedal Down" state. The attacker can exploit the distributed control features of ROS either to remotely terminate the RAVEN node/process (*/r2_control* process in Figure 5.14(a)) or to modify the parameters used by the RAVEN node/process (e.g., by overwriting the motor gain parameters on the ROS parameters server).

```
%time,field.header.seq,field.header.stamp,field.header.frame_id,field.level,field.name,field.msg,field.file,field.
function,field.line,field.topics0,field.topics1,field.topics2,field.topics3,field.topics4,field.topics5,field.topics6
1439666825432555469,16,1439666825432555469,,8,/r2_control,Sat Aug 15 14:27:05 2015
1439666826430307961,17,1439666826430307961,,2,/r2_control,[[     'C'  : toggle console messages ]]
1439666826430338686,18,1439666826430338686,,2,/r2_control,[[     'T'  : specify joint torque    ]]
1439666826430364462,19,1439666826430364462,,2,/r2_control,[[     'M'  : set control mode       ]]
1439666826430386548,20,1439666826430386548,,2,/r2_control,[[     '^C' : Quit                   ]]
1439666826432068877,21,1439666826432068877,,2,/r2_control,*** Ready to teleoperate ***
. . .
1439666826433024289,27,1439666826433024289,,2,/r2_control,Entered homing mode
1439666826433067777,28,1439666826433067777,,2,/r2_control,put usb board id --> 48
39666826433081565,29,1439666826433081565,,2,/r2_control,put usb board id --> 33
39666826433099352,30,1439666826433099352,,2,/r2_control,USB Board started ->
. . .
1439666826588993541,38,1439666826588993541,,2,/r2_control,Entered runlevel 1
439666826590013865,39,1439666826590013865,,2,/r2_control,    -> sublevel 2
439666827100040594,40,1439666827100040594,,2,/r2_control,Homing sequence initialized
. . .
1439666850089061637,133,1439666850089061637,,2,/r2_control,Joint 10 ready
1439666850102074039,134,1439666850102074039,,2,/r2_control,Entered runlevel 2
1439666850102146481,135,1439666850102146481,,2,/r2_control,definitely the right arm -- t2j
1439666850103034732,136,1439666850103034732,,2,/r2_control,definitely the right arm -- t2j
_descriptions,/r2_control/parameter_updates
1439666851862040119,137,1439666851862040119,,2,/r2_control,Entered runlevel 3
. . .
```

Figure 5.15. Sample data collected by logging messages published to the rosout topic. The state of the surgical robot (e.g. Homing, runlevel 2 = “Pedal Up”, and runlevel 3 = “Pedal Down”) at each cycle can be determined by analyzing the log.

The sudden termination of the RAVEN node/process in the middle of a surgical procedure stops the robot from responding to the surgeon’s console commands. The surgical team needs to restart the system and re-execute the RAVEN software on the system. If the remote unprotected connection to the ROS master can be preserved after each restart of the robot, this attack practically makes the robot unavailable to the surgical team due to multiple halts of the RAVEN process.

5.4.1.3. Fixing ROS Vulnerabilities

Previous work has discussed possible solutions to address the vulnerabilities in the ROS. In [217] a runtime verification framework for ROS, called *ROSRV*, is proposed to provide safety and security for the ROS master and application nodes. *ROSRV* adds a new node called *RVMaster* in ROS for intercepting all node requests to ROS master and monitoring all the messages sent between nodes in order to enforce desirable safety and security policies. For example, access control policies can be implemented in *ROSRV* by allowing only certain nodes to publish messages

only to certain topics or authenticating the nodes when they are requesting to register or connect to the ROS master.

5.4.2. Attacks Compromising Remote Diagnostic Mechanisms

In this section, we describe the potential vulnerabilities in the servers used for remote service diagnostics of commercial surgical robots that, if exploited, can (i) make the remote service diagnostics unavailable to all the surgical robots or (ii) compromise the integrity and confidentiality of data transferred to/from the robots, causing difficulty in forensic investigation of safety and security incidents.

As discussed before, the da Vinci surgical system is currently the only FDA approved robot for general use in minimally invasive surgery [13]. The da Vinci system has a feature called *OnSite* remote diagnostics which is designed for communication of a robot (the client) with a remote access server to facilitate status updates, log uploads, and remotely accessing the robot control system for pre- and intra-operative troubleshooting [196] (see Figure 5.16). From the publicly available information provided for enabling the *OnSite* feature on robots located in healthcare facilities [220], we found that the firewall on the surgical robot is configured to establish a SSL/TLS session based on PKI certificate authentication to communicate with the remote access servers (e.g., dvms-dv.davinci-onsite.com) using the HTTPS protocol. We performed an SSL Server Test using Qualys SSL Labs website [221] on one of the remote access servers (dvms-dv.davinci-onsite.com). A copy of the test results is available at [222]. Next, we discuss the possibility of two attack scenarios based on these test results.

5.4.2.1. Denial of Service Attacks

We found that the da Vinci remote diagnostic server supports secure client-initiated renegotiation in a SSL/TLS session (RFC-5746). This feature can be exploited to launch a Denial of Service (DoS) attack and threaten the availability of the server (attack A1 in Figure 5.16). Although the remote server requires a client certificate to initiate a secure communication with the robot, any client without a valid certificate can initiate multiple requests for renegotiation of a client certificate and cryptographic parameters (i.e., a combination of the TLS protocol version, PKI key length, and encryption mode during a SSL/TLS session). The remote access server can be overloaded when it receives a large number of renegotiation requests to perform

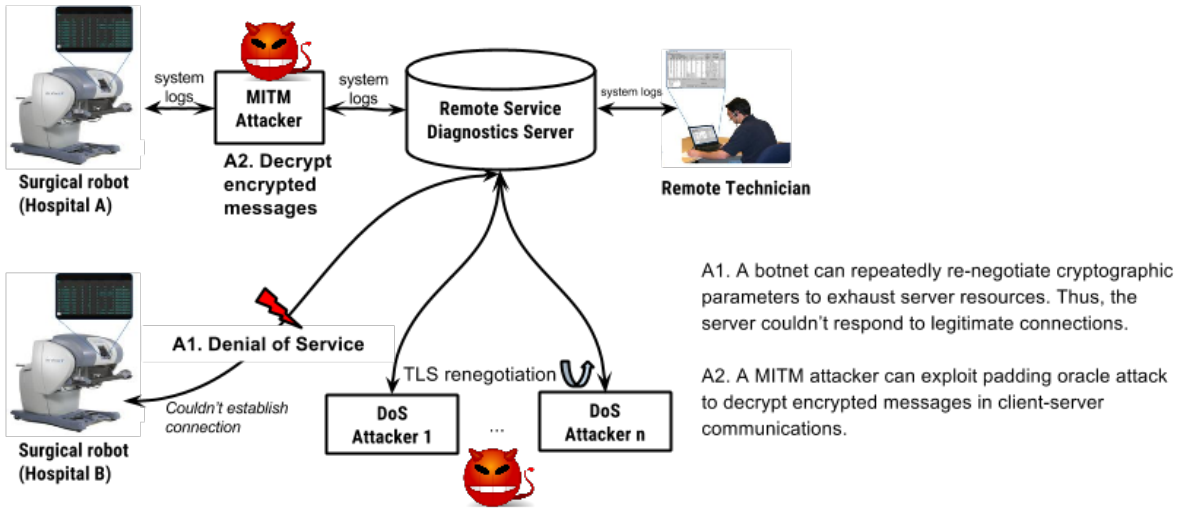


Figure 5.16.The attack scenarios targeting the remote access server used for performing service diagnostics on da Vinci robots during surgical procedures. We specifically simulated DOS attacks and man in the middle attacks exploiting the TLS renegotiation and POODLE vulnerabilities on our security test bed. Images are adapted and modified from [196].

computationally-intensive cryptographic functions. This might make the server non-responsive to legitimate remote assistance requests from surgical robots during procedure, when an unexpected error or emergency stop has occurred, and lead to prolonged procedure times and patient complications.

5.4.2.2. Man-In-The-Middle Attacks

The SSL test results further revealed that the remote diagnostics server (dvms-dv.davinci-onsite.com) is vulnerable to the Padding Oracle On Downgraded Legacy Encryption (POODLE) attack [222]. If an attacker has obtained access to the client certificate, through a man-in-the-middle (MITM) attack she can potentially force the use of the insecure cryptographic protocol SSL 3.0, instead of secure protocols such as TLS 1.x, to potentially decrypt the data transferred between the robot and the remote access server (attack A2 in Figure 5.16). If the client and the server exchange any sensitive information such as authentication cookies, the robot logs (e.g., robot operational state and robotic trajectories during a procedure), or status updates, such private information can be revealed to the attacker. However, this attack is highly dependent on the protocol used for communication between the surgical robot and the remote diagnostic servers.

5.4.2.3. Protection of Remote Diagnostics Mechanisms

The possible attacks on the remote access servers can be prevented by configuring the web server to properly implement protocol downgrades using TLS Fallback Signaling Cipher Suite Value (SCSV) [223]. The DDoS attack can be prevented by allowing only server-initiated TLS renegotiation to ask for client certificates and cryptographic parameters. Another solution is to restrict the maximum number of renegotiation requests in a period of time or the delay between two renegotiation requests allowed on the server.

5.5. Related Work

This section presents a summary of the previous work on security of teleoperated surgical robots and other safety-critical cyber-physical and process control systems as well as the recent studies and reports on real attacks to hospital networks. We also discuss the related work on safe path planning and navigation in dynamic robot environments.

5.5.1. Security of Teleoperated Surgical Robots

Previous work on security of telerobotic surgical systems mainly focused on network and communication-based attacks.

To show the actual risks of attacks to teleoperated surgical robots, Bonaci et al. [191] performed an experimental analysis of different cyber-security attacks on the communication between the surgeon's console and the robot on a RAVEN II platform. They evaluated the threats posed by attacks that modify or manipulate the intent of the surgeon or hijack control of the robot. Specifically, they measured the effect of lost, delayed, or modified packets sent from the surgeon's console on the performance of surgeons. Although these attacks caused jerky motions of the robotic arms and difficulties in performing the tasks by human operators, they did not cause unsafe operation of the robot control system. The robot's safety mechanisms detected and effectively stopped any possibly unsafe control commands.

Tozal et al. [187] used a novel information coding approach to design a Secure and Statistically Reliable UDP (SSR-UDP) protocol that ensures confidentiality and reliability of telesurgical communications in wireless environments. Lee et al. [188] proposed Secure ITP, a security enhancement to the Interoperable Telesurgury Protocol (ITP), introducing Transport Layer

Security (TLS) and Datagram TLS (DTLS) protocols for authenticating the master and slave devices as well as the surgeon and patient.

Most of the previous studies assumed that compromising a surgeon's control console or the robot control system is less likely because physical access to the system is prohibited through strict monitoring [190], [191]. Only Coble et al. [208] studied the possibility of compromising the robot software in unattended environments, such as the battlefield. They proposed the remote verification of system software and configuration files before execution, using remote software attestation.

Several recent security analysis reports have shown existing vulnerabilities in the hospital networks that can be exploited to gain remote elevated access to different medical devices in a hospital, including surgical robots (see Section 5.5.3). In this chapter, we described the anatomy of targeted attacks that potentially exploit the vulnerabilities in the hospital networks to get unauthorized access to the robot control system. To the best of our knowledge, this is the first demonstration of malicious attacks targeting the *control systems* of surgical robots.

5.5.2. Attacks on Process Control Systems

Attacks on the safety-critical cyber-physical or process control systems, such as smart power grid [224], [225], water plants [226], chemical plants [227], and automotive embedded systems [228], has been the subject of many studies. Most of the previous work focused on attack scenarios that directly target the physical system (e.g., smart meters in power grid infrastructure [224]) or the sensor measurements received from the physical system to corrupt the state of controller and state estimation process in the cyber-domain (false data injection attacks) [226], [227]. A few studies focused on control-related attacks that corrupt the control commands in industrial control systems [225], [226]. Lin et al. [225] demonstrated a class of control-related attacks on the SCADA (Supervisory Control and Data Acquisition) systems in power grid where the control fields in the network packets are modified in a legitimate format that cannot be detected by existing intrusion detection systems and can lead to catastrophic impact on the physical power system.

Both [225] and [227] use the dynamic models of process control system to estimate the state of physical system and consequence of control commands in order to detect safety-critical attacks. However, the cyber-physical systems studied in [225] and [227] are not constrained by the tight

real-time requirements such as those imposed on robotic surgical systems. For example, in the smart power grid the control commands are delivered to destination in several hundred milliseconds [225] or in the chemical reactor plants, the time for a human response to an attack before reaching to a unsafe state could be a couple of hours [227]. Therefore, the attack detection mechanisms can run in parallel with the system and detect malicious commands or measurements *after* their first appearance. The attack scenarios demonstrated in this chapter can impact the operation of surgical robot and safety of patient in just a couple of milliseconds, making it difficult for both automated mechanisms and human operators to respond in a timely manner. So the detection and response mechanisms in real-time cyber-physical systems should be optimized and deployed in such a way that can estimate and mitigate the impact of malicious commands *before* they even execute in the physical layer.

Further, in all previous studies the catastrophic impact of attacks and performance of detection mechanisms were evaluated in simulation or using theoretical models. In this work, we evaluated the accuracy of our dynamic model by comparing it to the actual robot using both attack-free and simulated attack scenarios. Another unique aspect of our work is that we demonstrate the actual implementation of malware and the phases for deploying the attacks on a real system. Most of the previous work only focused on theoretical representation of attack scenarios and did not discuss the practical implications and difficulties in implementing those attacks in the real system.

5.5.3. Attacks on the Hospital Networks

In the attack scenarios presented in Section 5.2, we assumed that attackers exploit one of the existing vulnerabilities in the hospital networks to get access to the telerobotic surgical systems, without being detected by regular security monitoring mechanisms, such as intrusion detection systems or remote software attestation techniques. Table 5.2 presents a summary of the recent reports on real attacks to hospital networks.

For example, TrapX Security Inc. recently discovered three targeted attacks on a hospital's network that passed through the protection of antivirus software, intrusion detection systems, and firewalls. In one case, the vulnerabilities in a blood gas analyzer was exploited to establish a backdoor to the whole hospital network, allowing the attackers to install a malware on the system and steal patient data records from the hospital. In another case, the attackers gained unauthorized

access to a clinic workstation, by stealing credentials of an employee visiting a malicious website and installing a malware in that machine [185].

In another recent study on a wide range of medical devices in several hospitals, researchers from Essentia Health discovered that the internal firewalls used for protecting surgical robots from external connections (see Figure 5.1) might crash upon running a vulnerability scanner against them and enable unauthorized access to the robot [180].

In addition, there have been several FDA recalls and adverse event reports to the FDA MAUDE database on non-targeted attacks on hospital networks in which malware or viruses infected medical devices such as imaging systems, causing interruptions in patient therapy [229], [230].

5.5.4. Safety and Reliability of Robotic Systems

In the context of safety and reliability of robotic systems, the works mostly related to ours are those on (i) collision avoidance using motion-planning algorithms and external sensors (e.g., sensitive skins or on-board vision) or (ii) fast collision detection and reaction mechanisms using adaptive control.

The previous works [231] – [233] used the comparison between nominal estimated robot states calculated based on dynamic models against the actual sensor measurements (e.g., joint encoder readings or torque commands) from the robot to rapidly detect collisions. In their case the estimated states were used as reference points (expected values in absence of collisions) for detecting deviations that have already occurred (detection after collision happened). But in our approach, we use the previously learned bounds on the nominal robot trajectories as the reference points to compare with the estimated states to *preemptively* detect the upcoming deviations. Our anomaly detection technique is similar to [234] where a model of normal execution of the robot is learned based on redundant information from multiple sources (e.g., as wheel encoder readings and localization algorithm output) to detect anomalies in the behavior of robot at runtime.

Robust and adaptive algorithms are proposed to control the robot in case of actuator faults [235] or react to collisions by safely driving the robot away from the human and collision area [231], [236] or regulating the force at the contact point [232].

The related works in this domain mainly focus on industrial, autonomous, or mobile robots and their safe interaction with the humans. To the best of our knowledge, no previous work addressed

the challenges in design of real-time detection and recovery strategies for faults, abrupt jumps, or collisions in non-autonomous teleoperated robotic manipulators, such as surgical robots.

5.6. Conclusions

In this chapter, we described the anatomy of a family of targeted attacks against the *control systems* of teleported surgical robots. These attacks are deployed by installing a self-triggered malware that measures the leaked information on the operational state of the robot to infer a critical time during surgery to inject malicious control commands to the motor controllers or to halt the robot control process. An important characteristic of these attack scenarios which may make them more favorable to malicious parties compared to other random attacks that they are stealthy in nature and hard to distinguish from system failures/misbehavior caused by unexpected physical malfunctions or unintentional human mistakes. This makes the forensic investigation of incidents more complicated. The detection and mitigation of such security exploits also becomes more difficult, requiring the understanding the semantics of robot control and communications as well as concurrent monitoring of both cyber and physical system components. We demonstrated these attacks on the RAVEN II surgical robot and experimentally evaluated the impact of the attacks on the operation of the robot control system and safety of patients.

We presented a model-based analysis framework that can estimate the consequences of control commands through real-time computation of the system's dynamics and detect the unsafe commands before they are executed in the physical system. Our experiments demonstrate that: (a) injecting malicious commands to the control software can lead to *unforeseen and abrupt jumps* of a few millimeters in the robot manipulators within only a few milliseconds or unavailability of the system due to *unwanted transition to a halt state,* and (b) our dynamic-model based analysis framework can detect malicious commands and potentially mitigate their impact before they manifest in the physical system, with an average accuracy of 92%.

Further, the ideal location for insertion of the proposed safety mechanisms is at the lower layers of control structure, at the latest computational step before executing the commands on the physical system. This scheme reduces the possibility of TOCTTOU exploits at software layer and mitigates any exploits that skipped or survived previous layers of safety checks, and prevents any possible adverse consequences in the physical system.

We also demonstrated other possible ways that the availability of surgical robots can be compromised on a wider scale, specifically discussing vulnerabilities in the Robotic Operating System and the servers used for remote diagnostic services during surgery.

This study shows the importance of designing detection and mitigation mechanisms that combine understanding the semantics of both software and physical components to detect and prevent the attacks and distinguish them from accidental failures in the safety-critical cyber-physical systems.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

This dissertation described a data-driven approach to assessment and design of resilient medical cyber-physical systems. The presented approach is based on the analysis of real data on safety incidents involving medical devices as a basis for developing tools for (i) gaining deeper understanding of the causes of incidents and providing statistically confident measures of their impacts, (ii) validating the resiliency of safety mechanisms in the presence of accidental failures and malicious attacks, and (iii) detecting safety hazards in real-time and mitigating adverse impacts on the physical system and patients.

To address the challenges in analysis of medical device incidents, we presented MedSafe, a toolset for automated analysis of structured and unstructured data on recalls and adverse events from the publicly available FDA databases. By combining techniques from natural language processing, machine learning, and accident causality modeling, MedSafe enables characterization of the safety issues associated with from computer failures in medical devices, in terms of fault classes, failure modes, and recovery actions taken by the manufacturers, and measures their cost in terms of severity of hazards for patients, numbers of devices affected on the market, and number of device repairs or removals. MedSafe adverse event analysis is driven by a novel ontology model based on the hierarchical control structures used in system-theoretic accident causality analysis. The proposed ontology model formalizes the semantic interpretation of incident descriptions and extraction of important safety-related features (e.g., faulty device conditions, unsafe operator actions, and interactions among operators and the system) from the adverse event narratives.

We evaluated MedSafe by performing large-scale studies on more than 18K FDA recall records related to a variety of medical devices and over 10K adverse events reported on robotic systems used in minimally invasive surgery. These studies were the *first large-scale automated analyses* of the FDA recalls and adverse events data, and MedSafe achieved an average accuracy of over 90%. Our analysis revealed that although software remains the major cause of failures in computer-

based medical devices, hardware, battery, and I/O failures have much larger impact in terms of number of recalled devices and cost of device removal/repairs. We identified several examples of safety-critical medical devices that either were designed without proper identification and handling of safety hazards or had safety mechanisms that were not validated adequately. By analysis of adverse events in robotic surgery, we found that if an adverse event happened during a procedure, there was about a 24% chance of negative impact on patients or interruption in progress of the surgery (to troubleshoot problems by manual system resets, convert to non-robotic techniques, or reschedule the procedure). Our analysis showed the need for developing improved safety mechanisms for preemptive detection and mitigation of hazards and safety-training programs that prepare surgeons for handling safety-critical scenarios.

To assess system resiliency in the presence of potential safety hazards and security attacks, we developed a safety hazard injection framework that can validate the system's safety mechanisms by emulating potential safety hazards caused by accidental failures or malicious threats targeted at different layers of the system control structure. Using this framework, we identified several vulnerabilities in the safety mechanisms of the RAVEN II surgical robot. We also applied this framework to the simulation of realistic hazard scenarios (identified through causal analysis of FDA adverse events) in a virtual environment, which can be used for simulation-based safety training of robotic surgeons.

We further studied the resiliency of surgical robots to safety hazards caused by malicious attacks. We introduced a family of cyber-physical attacks that target the control system of teleoperated surgical robots and can cause the system to move to an unwanted state, damage the physical system, or harm the patient in the middle of surgery. Those attacks were demonstrated by development of a self-triggered malware on the control system of the RAVEN II surgical robot. Our implementation of attacks exploited the vulnerabilities in the robot control software and operating software to inject malicious control commands at a critical time during robot operation. We asserted that preemptive detection of those attacks requires continuous monitoring of both cyber and physical components and prediction of the consequences of commands executed at different layers of the system control structure.

We presented a dynamic-model based analysis framework that can preemptively detect the adverse consequences of malicious control commands (e.g., abrupt jumps of robotic arms) through real-time computation of the system's dynamics and estimation of the next system states. Our

experiments on the RAVEN II robot showed that the presented framework can detect and potentially stop the malicious commands before they impose adverse impact on the physical robot and patient, with an average accuracy of 92%.

6.2. Future Work

There remain many opportunities for future work based on the research described in this dissertation. The following are a few directions that can be explored in the future:

- **Systems-theoretic data collection:** We presented MedSafe for automated analysis of publicly available data on medical device adverse events from the FDA MAUDE database. However, analysis of the adverse event data is still a challenging task, because of underreporting and incomplete, inaccurate, and inconsistent information manually entered by the manufacturers and volunteer reporters. The system-theoretic accident causality modeling techniques used in this dissertation can be further applied to collection of accurate information on medical device adverse events. For example, the STAMP causality model can be used to derive the design and strategic placement of data recording (logging) mechanisms that automatically collect and process important safety information (e.g., human operator actions, system component states, patient status, and the interactions among them as modeled by each control loop) from different layers of the system control structure. The data (or event) recorders ("black boxes") have been widely deployed in other safety-critical systems, such as automobiles and airplanes, and have greatly facilitated the investigation of accidents [237], [238]. Real-time collection and fusion of such data can further provide opportunities for design of monitors that make quantitative measurements of the system's safety and mitigate impeding hazards in a timely manner.
- **Robust safety engines for timely recovery from safety hazards:** Our analysis of the FDA data showed that although the medical devices are often designed with safety mechanisms that detect system failures and put the system into a safe state, their diagnostic mechanisms are not comprehensive enough to correctly identify the causes of safety hazards and perform timely and effective recovery actions. The main reasons are the complex nature of incidents, the difficulty of accurately modeling the causality relationships, and uncertainties in operator actions, effectiveness of safety mechanisms, and patient health status. Future research can focus on design of safety monitoring engines that combine real-time

measurements of the operator actions, physical system state, and patient status with knowledge of the dependencies between the causal factors that lead to safety hazards; the goal would be to decide on the most effective action for hazard mitigation based on the predicted risks. One possibility is to use time-varying graphical probabilistic models (such as Dynamic Bayesian Networks (DBNs) [239]) to construct an inference engine that can estimate the likelihood of impending safety hazards by taking into account the uncertainties of operator actions, robotic software and hardware state (in the presence of faults), and patient status. Previous work has applied time-varying graphical probabilistic models to reliability modeling [240], safety decision-making [241], real-time detection and prediction [242], [243], and preemptive detection of security attacks [244], [245]. Appendix C presents our previous work on design of robust patient monitoring devices by using a non-probabilistic approach that combines real-time analysis and fusion of biosignals collected from the patient with online monitoring of activity signals from device functional units. Our proposed solution enables detection of early signs of health deterioration as well as computational failures and dynamic recovery from failures at low performance and energy overheads [246] – [249].

- **Application to other safety-critical cyber-physical systems:** The safety and security assessment methods presented in this dissertation can potentially be applied to a broader class of embedded and cyber-physical systems that involve humans in on-line decision-making and control, including the electric power grid and transportation systems. We discussed several challenges in detection and mitigation of malicious attacks on control systems of surgical robots that can be generalized to other safety-critical cyber-physical systems [250]. Example challenges that provide opportunities for further research include:
  (i) Existence of the intrinsic TOCTTOU gap between the cyber and physical layers that can be exploited to compromise the control commands and feedback.
  (ii) Lack of mechanisms for simultaneous monitoring of both cyber and physical system components and states.
  (iii) Real-time constraints in design of control systems that impose restrictions on designing comprehensive monitoring and diagnostic mechanisms.
  (iv) Difficulty of diagnosing the causes of safety hazards and distinguishing accidental failures from (human-induced) malicious attacks.

APPENDIX A
UNDERREPORTING IN DATA ANALYSIS

The underreporting in data collection and analysis is a fairly common problem in social sciences, public health, criminology, and microeconomics. It occurs when the counting of some event of interest is for some reason incomplete or there are errors in recording the outcomes. Examples are unemployment data, infectious or chronic disease data (e.g., HIV or diabetes), crimes with an aspect of shame (e.g., sexuality and domestic violence), error counts in a production processes or software engineering, and traffic accidents with minor damage [251]. An estimated prevalence of events based on the incomplete counts is likely to be smaller than the true proportion of events in the population. Several inference techniques based on binomial, beta-binomial, and regression models have been proposed for estimating the actual count values [252]. However, in all those techniques the reporting probability (underreporting rate) is assumed to be a constant parameter over time that is estimated based on the sample counts.

A very similar problem exists in preliminary or pilot clinical investigations, epidemiological surveys, and longitude studies where the objective is to estimate any possible clinical effect of a treatment or prevalence of a particular disease in a population of patients, but the prevalence of events can only be estimated by selecting a sample of patients from the population [253].

In all these situations, the prevalence of the events are estimated based on a random sample of events from the population, under the assumption that the sample set contains the same characteristics and distributions of the actual population, including those of the underreported and missing cases.

Furthermore, it is often required to perform a sample-size calculation based on confidence intervals in order to provide a precise estimate with a large margin of certainty and to make sure that the estimated proportion is close to the actual proportion with a high probability [253]. Confidence intervals for the proportions estimated based on samples from large populations and finite populations can be calculated by using the normal approximation to the binomial distribution as follows:

For large populations:

$$p \pm z_{1-\alpha/2}\sqrt{\frac{p(1-p)}{N}}$$

For finite populations:

$$p \pm z_{1-\alpha/2}\sqrt{\left[\frac{p(1-p)}{N}.\frac{N_{Population-N}}{N_{Population}}\right]}$$

where N is the size of sample, $p = \frac{r}{N}$ is the estimate of the proportion of events of interests in the sample and $N_{Population}$ is the size of population in case of finite populations [253].

In this study, we estimated the prevalence of adverse events by making sure that we have a significantly large enough number of samples to provide confident estimates. Our estimations are obtained under the assumption that the characteristics and distributions of the observed events are not significantly different from those in the actual population and would not significantly change after including the underreported cases. We are currently investigating the extension of the proposed inference techniques in [250], [252] to estimate the actual number of adverse events by considering variable reporting probabilities over time.

APPENDIX B
EXAMPLE ADVERSE EVENT REPORTS

MAUDE Report 2567858

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=2567858

Event Date: 04/09/2012	**Event Type:** Death	**Patient Outcome:** Death

Event Description: It was reported that approximately 45 minutes into a da vinci si procedure when the assistant at the patient side cart (psc) performed a tool change to switch the instruments in the patient side manipulator (psm) arms, the patient's aorta was punctured. The surgeon made the decision to convert to traditional open surgical techniques to complete the planned surgical procedure. The patient was reported to be in stable condition right after the open surgical procedure, however, the patient expired the following day. The site has been contacted for additional information, however, no additional information has been forthcoming.

Manufacturer Narrative: The isi clinical sales representative followed up with the assistant rn and the rn indicated that the patient side assistant had not received training by the site on how to perform a guided tool change. Based on the information provided, the puncturing of the patient aorta is the result of an incorrect instrument change. The surgeon had requested that instruments located on the patient side manipulators (psm) arms be swapped, such that the instrument on psm arm 1 would be switched with the instrument located on psm arm 2. It is not clear whether the assistant nurse used a guided tool change to safely change instruments or whether the nurse used the arm clutch to insert the instrument manually. It is unclear whether any malfunction of the system occurred and the exact mechanism which resulted in puncturing of the aorta. The site has been contacted for additional information and once this information has been received, a follow up report will be sent to the fda. The da vinci si instructions for use (ifu) specifically states: warning: the instrument may not be immediately visible when being moved from the cannula into the patient. Use appropriate caution when manually inserting instruments into the patient.

MAUDE Report 1891889

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=1891889

Event Date: 08/26/2010	**Event Type:** Injury	**Patient Outcome:** Other

Event Description: It was reported that prior to starting a da vinci si hysterectomy procedure, the site received a vision system error. The isi representative on site began to troubleshoot and found the left eye on the surgeon side console was not working. The surgeon and or staff were notified of the system status; however, the surgeon decided to perform the case with one eye working in 2d vision. During the procedure the patient's ureteral was injured and a urologist was called in to perform the repair and place a stent. At this time the case was converted to traditional open surgical techniques to repair the patient's ureteral and complete the planned procedure. The patient was required to stay in the hospital an additional 24-48 hours due to the open incision as well as the stent removal. On (b)(4) 2010, intuitive surgical received maude event report (b)(4) for this event.

Manufacturer Narrative: The investigation conducted by field service engineering found the camera cable had a bad left eye channel. The camera cable connects the camera to the system's vision cart, which then transmits the image to the surgeon side console. The system was repaired by replacing the defective camera cable. As of november 4, 2010, there have been no reported recurrences of the issue at this hospital.

MAUDE Report 1760256

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=1760256

Report Date: 06/17/2010	**Event Type:** Injury	**Patient Outcome:** Disability

Event Description: It was reported that during a da vinci s prostatectomy procedure performed on (b)(6) 2008, the surgeon experienced interference in the left eye image viewed from the surgeon's console. The surgeon switched the image from 3-dimensional (3-d) to 2-dimensional (2-d) to proceed with the procedure. The planned surgical procedure was successfully completed and no patient harm, adverse outcome or injury was reported. The customer-reported-event does not in itself constitute a reportable event, however, on june 17, 2010, isi received a legal summons and complaint filed by the patient, alleging that due to the amount of time added by completing the surgery in 2-d (the surgery lasted over 8 hours) the patient sustained a severe and permanent right leg injury, specifically right calf compartment syndrome. The complaint further alleges that the patient has undergone numerous and extensive medical procedures, including surgery and therapy.

Manufacturer Narrative: The investigation conducted by the field service engineer concluded that the vision issue experienced by the customer was associated with a faulty camera cable. The camera cable connects the camera to the system's vision cart, which then transmits the image to the surgeon's console. The system was repaired by replacing the affected camera cable. The camera cable was returned to the original equipment manufacturer (oem) for evaluation. The oem observed that the left channel of the camera cable had a loose connection, thus causing the vision issue experienced by the customer. The da vinci s surgical system user's manual explicitly states that, environmental or equipment failures may cause the da vinci s system to become unavailable. The surgical team should always have backup equipment and instrumentation available, and be prepared to convert to alternative surgical techniques.

MAUDE Report 2476271

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=2476271

Report Date: 02/02/2012	**Event Type:** Injury	**Patient Outcome:** Required Intervention

Event Description: It was reported that during a da vinci hysterectomy procedure, after the bipolar cord was connected to the electrosurgical unit, energy was released from the bipolar instrument. As a result, an injury to the patient's bowel occurred. The bowel damage was secured by suture and the planned surgical procedure was completed. No additional harm was reported.

Manufacturer Narrative: The hospital has been contacted multiple times for further details and information concerning this incident. To date, no response has been received. As a result, the root cause of the reported event cannot be determined. A follow-up medwatch report will be submitted if additional information is received. On (b)(6) 2012, the hospital provided intuitive surgical a copy of medwatch uf/importer report (b)(4), which contained the following information: clinical engineering analysis: when connecting the bipolar cord to the esu, apparently the nurse inadvertently plugged the cord into the monopolar jack instead of the bipolar jack. Since the tips of the bipolar forceps were closed, this triggered the monopolar cut mode of the esu and it began delivering energy. Staff noticed the esu tone and smoke visible on the video display and immediately shut off the power to the esu. The monopolar jack on the esu has three holes. Connecting the two outer holes will activate the cut mode, as this is how the normal pencil switch works. The disposable bipolar cord has individual banana plugs on the machine end, which makes it possible to plug into the wrong jack. If the machine end had a single molded connector, the spacing of bipolar pins would prevent plugging into a monopolar jack. Per the additional information provided above, intuitive surgical concluded that the issue experienced by the hospital was due to improper connection of the bipolar instrument to the electrosurgical unit (esu). The instruments and accessories user manual specifically states: caution: please refer to the individual esu manufacturer's operator's manuals for operating instructions. Set the esu to bipolar output. Set the power as low as possible to achieve adequate hemostatis. Warning: do not use bipolar instruments with a monopolar source output as this may cause damage to the instrument and harm to the patient or medical personnel.

MAUDE Report 2494890

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=2494890

Report Date: 02/16/2012	**Event Type:** Injury	**Patient Outcome:** Other

Event Description: It was reported that during a da vinci s coronary artery bypass graft (cabg) procedure, arcing from the micro bipolar forceps instrument was observed when the surgeon was not applying cautery energy. The site contacted isi technical support engineering (tse) for troubleshooting assistance, however, the site declined support, as the surgeon did not have time to trouble shoot the issue with the tse. The planned surgical procedure was completed and no patient harm adverse outcome or injury was reported.

Manufacturer Narrative: On (b)(6) 2012, intuitive surgical received uf/importer report (b)(4) from the fda. The event details are provided below: event description: patient for a cabg with a da vinci robot. Surgeon attempted to use the da vinci bipolar forceps, but they did not work. Circulator checked the electrosurgical unit (esu and noted that ground pad (return electrode for esu was not plugged in. Ground pad connected. Surgeon stated there was no noise and the foot pedal was not depressed, but as soon as the surgeon touched the diaphragm with the bipolar forceps, the machine buzzed and the diaphragm had a small burn. Circulator checked the esu and noted that the forceps cable was plugged into two of the three monopolar 2 sockets. Bipolar was plugged into bipolar sockets, machine functioned normally throughout the rest of the case. Patient had two (2) ground pads on, unsure which was the actual lot number involved in event. Based on the additional information provided in the site's u/f report (b)(4) indicating that the patient's diaphram was burned, on (b)(6) 2012, isi contacted risk manager, (b)(6) to advise that during isi's initial investigation into the reported event, it was reported by the site's da vinci coordinator, (b)(6) initial reporter) that no patient harm, adverse outcome or injury occurred. (b)(6) indicated that contrary to the initial information provided, a burn to the patient's diaphram occurred, however, no repair of the affected area was required, the planned surgical procedure was completed, and there was no report that the patient experienced any post-operative complications. She does not know if the patient has had to return to the hospital due to any post-operative complications related to the reported event. As indicated in isi's initial medwatch report submitted to the fda on (b)(4) 2012, isi's investigation into the reported event found that the issue experienced by the site was due to improper connection of the micro bipolar forceps instrument to the electrical surgical unit (esu). The instruments and accessories user manual specifically states: general precautions and warnings o please refer to the individual esu manufacturer's operator's manual for operating instructions. Set the esu to bipolar output. Set the power as low a possible to achieve adequate hemostasis - do not use bipolar instruments with a monopolar source output as this may cause damage to the instrument and harm to the patient or medical personnel system functional testing performed by the fse found that the system functioned within specification.

The investigation conducted by field service engineering (fse) found that the issue experienced by the site was due to improper connection of the micro bipolar forceps instrument to the electrical surgical unit (esu). The hospital's biomedical department indicated to the fse that during the surgical procedure, the micro bipolar forceps instrument was found to be incorrectly plugged into the monopolar connection output on the electrical surgical unit (esu). The site immediately resolved the issue by connecting the instrument to the bipolar connection site output on the esu as indicated in our ifu. The instruments and accessories user manual specifically states: general precautions and warnings o please refer to the individual esu manufacturer's operator's manual for operating instructions. Set the esu to bipolar output. Set the power as low a possible to achieve adequate hemostatsis o do not use bipolar instruments with a monpolar source output as this may cause damage to the instrument and harm to the patient or medical personnel system functional testing performed by the fse found that the system functioned within specification. On (b)(4) 2012, isi followed up with the initial reporter at university of medical center and he confirmed that no patient harm, adverse outcome or injury occurred. The patient tolerated the planned surgical procedure well and has not returned to the hospital due to any post operative complications. As of (b)(4) 2012, there have been no reported recurrences of the issue at this hospital.

MAUDE Report 3024317

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=3024317

Report Date: 02/11/2013	**Event Type:** Injury	**Patient Outcome:** Other

Event Description: It was reported that during a da vinci si prostatectomy procedure, when the surgeon activated bipolar cautery energy from the footpedal on the surgeon side cart, cautery energy from the monopolar curved scissors (mcs) instrument was activated causing a burn to the patient's bowel.

Manufacturer Narrative: On (b)(4) 2013 an inspection of the site's da vinci si system conducted by an isi field service engineer (fse) was unable to replicate the cautery issue experienced by the site. Functional testing performed on the site's system by the fse found that the site's da vinci si surgical system functioned within specification. On (b)(4) 2013 isi contacted the isi clinical sales representative (csr) who reported this event. The csr indicated that she met with the operating room director and operating room scrub technician at the site on (b)(4) 2013 and the operating room scrub technician indicated to her that automatic activation of cautery energy was due to the surgical staff had incorrectly connected the electrical surgical unit (esu) cables into the improper receptacles on the esu. The scrub technician indicated that the esu connections were corrected, and the issue was resolved. The planned surgical procedure was completed. A general surgeon at the hospital was consulted to access the damage to the patient's bowel burn. The general surgeon's assesment of the burn to the patient's bowel determined that no repair of the affected area was required. On (b)(4) 2013, isi contacted the surgeon who performed the surgical procedure. The surgeon indicated that while he was retracting the patient's bowel using the maryland bipolar forceps instrument, he heard the alarm from the esu and immediately stopped using the maryland bipolar forceps instrument. The surgeon indicated that he had not touched the footpedal and that the instrument activated when he had not pressed the footpedal. The surgeon indicated that the patient sustained a small burn injury to the sigmoid colon and that he attempted to consult with a general surgeon concerning the patient's injury; however, there was no general surgeon available at that time. The surgeon indicated that he assessed the damage to the patient's bowel and he determined that the damage did not require any repair. The surgeon indicated that the patient remained in the hospital an additional day for monitoring and that the patient was discharged from the hospital. The surgeon indicated that the patient is recovering well and has not returned to the hospital due to experiencing any post-surgical complications as a result of the reported event. Intuitive surgical's instruments and accessories user manual precautions and warnings indicate: caution: please refer to the individual esu manufacturer's user manual for operating instructions. Warning: do not use monopolar instruments with a bipolar source output as this may cause damage to the instrument and harm to the patient or medical personnel. This report does not admit that the report or information submitted under this report constitutes an admission that the device, intuitive surgical or intuitive surgical employees, caused or contributed to the reportable event.

MAUDE Report 2632716

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=2632716

Event Date: 04/19/2012	**Event Type:** Injury	**Patient Outcome:** N/A

Event Description:"pt in operating room for robotic laparoscopic hysterectomy; surgeon documented grossly enlarged uterus with multiple fibroids. As surgeon was bivalving the uterus, he saw a flash of blood coming from the left side of the pt and immediately recognized vascular injury. The robot was immediately undocked and procedure was converted to open procedure. The vascular surgeon responded immediately to repair the pt's left external iliac artery, which he described as "charred" due to the cautery injury. Operating room staff reported small tear in the plastic that covered the metal end of the monopolar scissor. The pt was stabilized after surgical repair and administration of blood products and transferred to the intensive care unit post-operatively. She remains stable and was transferred to the surgical unit the following day. The pt was discharged home (b)(6) 2012".

Manufacturer Narrative: The mcs instrument was returned and evaluated. Per the customer reported complaint (with the following clarification) " the defect was a tube extension dislodged from the main tube". The entire tube extension wall is circumferentially broken at the base of the axial keys. As a result, the tube extension can be rotated 360 degrees. The surface of the reinforcement tube extension exhibited burnt molded material, indicating unintended arcing may have occurred. Engineering concluded that the dislodged tube at the distal end is likely due to overloading at the distal end which led to a path for unintended arching. The mcs tip cover accessory was not returned. On (b)(4) 2012, isi contacted risk mgr, (b)(6) at (b)(6) and she indicated that the pt had a large uterus which required that the surgeon cut the pt's uterus in half to allow for removal. She indicated that the surgeon did not observe arcing from the instrument however; there was an excessive amount of smoke in the surgical area when the injury to the pt occurred. Since she stated that a vascular surgeon was immediately called to repair the affected vessel and during repair of the vessel, the vascular surgeon noted that it exhibited charring, thus it was determined that an arcing event had occurred. Per (b)(6), the surgeon did not observe that the mcs instrument broke during the surgical procedure, however, examination of the tip cover accessory by the surgical staff found that it had a hole; however, she does not know the status of the tip cover accessory. (b)(6) indicated that the pt is recovering well and has not returned to the hospital due to experiencing any post-operative complications. The instruments and accessories user manual specifically states: general precautions and warnings: handle instruments with care. Avoid mechanical shock or stress that can cause damage to the instruments. Do not use an instrument to clean debris from another instrument intraoperatively. This may result in damage to the instruments or other unintended consequences, such as disconnection of the instrument tip. This mdr is being submitted for retrospective activity performed relating to field action number 2955842-051613-005 to investigate micro-cracks on the monopolar curved scissors instrument. These types of micro-cracks will not lead to mechanical failure of the instrument; however, there is a potential for insulation failure after reprocessing, resulting in a pathway for electrosurgical energy to leak to tissue and potentially cause unintended injuries. The location of theses micro-cracks is confined to 2 cm of the distal end of the instrument shaft. This instrument was inspected as part of this retrospective activity and found to contain cracks on the instrument main tube.

MAUDE Report 3473388

http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=3473388

Report Date: 10/18/2013	**Event Type:** Injury	**Patient Outcome:** Required Intervention

Event Description: On (b)(6) 2013, the user facility contacted an intuitive surgical, inc. (isi) clinical sales representative (csr) during a da vinci myomectomy procedure and requested her to come in because multiple instruments kept breaking. When the csr arrived, she noted that the port placement was incorrect for the type of surgical procedure and the vision was obscured, not allowing the instrument tips to be viewed. The csr stated that the surgeon was resistant to her recommendations but finally agreed to move the camera port for a better view of the instruments. According to the csr, the surgeon repeatedly over rotated the master tool manipulator (mtm) and kept losing view of the instrument's tips. The master tool manipulator refers to the master controllers which provide the means for the surgeon to control the instruments and endoscope inside the patient from the surgeon side console. The instruments were being pushed against the myoma with enough force that it was causing the instruments to break. The csr stated she witnessed the instrument cables breaking on a maryland bipolar instrument. The csr observed that the view of the monopolar curved scissors (mcs) instrument tips was lost and the surgeon was engaging the mcs instrument as if he was cutting tissue and the patient sustained a possible artery nick in the right pelvic side wall. The da vinci procedure was then converted to an open surgical procedure. The csr was unable to obtain information regarding the broken instruments that were used prior to her arrival. The following day, an isi field service engineer (fse) went onsite and did not find any issues with the da vinci system and verified that the da vinci system was ready for use. On (b)(6) 2013, the user facility reported damage to 4 different instruments that were used during this reported event. On (b)(6) 2013, isi spoke with the csr. The csr stated that the patient's right external iliac vein was transected. She was unable to provide additional information regarding the injury and details regarding the reason why the surgeon converted the procedure to an open surgical procedure. She also stated that the surgeon had not performed any da vinci surgeries between (b)(6) 2012 and (b)(6) 2013. The csr reported that she had offered the surgeon supplemental training sessions prior to the procedure but they were not undertaken. On (b)(6) 2013, an isi clinical consultant spoke with the bed-side assistant surgeon to the da vinci surgeon performing the case. He confirmed that the patient's right iliac vein was lacerated and the procedure was converted to an open surgical procedure to repair the injury. He confirmed that the da vinci surgeon had difficulty operating with the da vinci system. He recalled there was over rotation of the master tool manipulator (mtm) and some degree of frustration in keeping all instrumentation in view. It was then that injury occurred, although his attention at the exact moment of the damage was on a different part of the screen. He did not think that the injury happened out of view of the camera.

Manufacturer Narrative: On (b)(6) 2013, the user facility reported damage to 4 different instruments that were used during this reported event. Review of the system log confirmed that the reported instruments were 4 of the 5 instruments that were used during the reported da vinci procedure. The 4 instruments involved with the reported event have been returned to isi and evaluated. Failure analysis investigations noted the following findings: instrument 1: monopolar curved scissors (mcs) (part 420179, lot m14130619-891): findings: tube extension was found broken and was missing a piece at the distal end. Instrument 2: monopolar curved scissors (part 420179, lot m14130619-047): findings: tube extension was found broken and was missing a piece at the distal end. Instrument 3: permanent cautery spatula (part 420184-06 , lot m10120523-783): findings: broken ceramic sleeve, heavy biodebris and black burnt or char marks residing around the spatula, and an uneven piece of the ceramic sleeve was missing, exposing the shaft of the spatula. There was also a derailed yaw cable at the instrument's wrist and both pitch cables were broken. Please reference mdr with patient identifier (b)(6). Instrument 4: maryland bipolar forceps (part 420172-07 , lot m10130819): findings: broken pitch cable at the proximal clevis hub. Please reference mdr with patient identifier 700108065. Investigation noted that the damage found on the 2 mcs instruments and permanent cautery spatula are likely due to misuse or mishandling. The instruments & accessories instructions for use (ifu) specifically states: handle instruments with care. Avoid mechanical shock or stress that can cause damage to the instruments. Based on the provided information, isi has not determined the root cause of the intra-surgical complications experienced by the patient. If additional information is received a follow up medwatch report will be submitted to the fda. This complaint is being reported due to the following conclusion: the patient sustained an injury during a da vinci surgical procedure and the procedure was converted to an open surgical procedure.

APPENDIX C
DEVICES FOR RESILIENT PATIENT MONITORING

Driven by our study on the recalls of safety-critical medical devices, we proposed a new patient monitoring system that can potentially recover from the common types of computer-related failures reported for the patient monitoring systems.

C.1. Computer Failures in Medical Monitoring Devices

In on our study on safety-critical failures reported to the FDA over 2006–2013, we specifically found ten safety-critical computer-related recalls related to physiological patient monitoring and arrhythmia detector devices, which affected around 38,394 devices on the market. Additionally, a total number of 359 adverse events, including four deaths, 79 injuries, and 276 malfunctions were reported for these devices [2].

Based on the example of recalls and adverse events reported to the FDA, we identified four sources of failures in medical monitoring devices: Data Errors, Algorithmic Inadequacies, Hardware Errors, and Software Errors. On one hand, the external environmental changes, physical perturbations, and sensor failures could cause the delivery of erroneous data inputs to the device and lead to improper functioning and incorrect results. For instance, the measured signals could get lost, noisy, or corrupted because of the failure of sensor nodes, their intrinsic noise, or electromagnetic interference. The patient's movements or changes of physical activities may also cause motion artifacts or deviated normal bounds of measured signals. On the other hand, internal faults that occur within the computational engines of the device, such as algorithm inadequacies (due to incorrect specifications), software bugs and hardware (transient or permanent) faults can also lead to safety-threatening detection inaccuracy and delay in the results, or even failure of the system. Table C.1 shows these failure categories along with their description and example recalls or adverse events reported to the FDA.

This chapter contains material from the works [246] – [249] coauthored with Q. Li, M. U. Saleheen, Z. Jin, C. D. Martino, Z. Kalbarczyk, and R. K. Iyer, copyrighted by IEEE.

Table C.1. Failure categories and example recalls and adverse events for medical monitoring devices.

Failure Source	Description	Example Recalls and Adverse Events		
		Report	Report Summary	Failure
Data Error	Erroneous input data streams due to noise, artifacts, or missing samples	MAUDE 2154693	Philips INTELLIVUE X2 Portable Patient Monitors have false asystole alarms at a much more frequent rate since the installation of additional wireless laptops in CICU, because of possible RF interference from increased level of radio frequency activity in the CICU	False Alarms
Algorithm Inadequacy	Algorithm not effective or inapplicable for a specific patient or medical condition	MAUDE 1614824	GE Healthcare APEX PRO FH Telemetry Monitoring Systems did not recognize a patient's telemetry rhythm and did not alarm a series of ventricular fibrillation events, leading to patient death. ECG (electrocardiogram) signal may have failed to meet necessary criteria for the arrhythmia algorithm.	Missed Detectio n
Hardware Error	Errors caused by transient or permanent hardware faults	Recall Z-2030-2009	Medtronic Physio-Control LIFEPAK CR Plus Defibrillator/Monitor had a short circuit in one of the relays on the analog printed circuit board that affected the ECG amplitude, causing the device to not analyze the ECG rhythm correctly and not delivering therapy.	Missed Therapy
Software Error	Errors due to software bugs and transient errors	Recall Z-2168-2011	Philips FloTrak Elite module used in NM3 is a multi-parameter patient monitor calculates two displayed respiratory parameters incorrectly (higher than actual) due to errors in the system software.	Incorrect Results

Given the criticality of application to the human life, medical monitoring systems have to be resilient in both accurate and timely delivery of results, despite the changes in the patient and environment and even in the face of accidental errors. Toward that objective, we specifically identify three main challenges for design of resilient medical monitoring devices:

- **Accuracy:** real-time analysis of physiological signals with low false positive/false negative rates.
- **Adaptability**: adaptation to patient-specific physiological characteristics, diagnostic needs, and different application scenarios.
- **Availability**: dependable monitoring that is resilient to unexpected artifacts and accidental errors.

C.2. Related Work

A limitation of existing patient monitoring systems is their fixed functionality and lack of functional adaptability. These devices are usually designed for analysis of only a certain type of physiological parameters associated with a specific medical condition, and cannot support multi-parameter analysis techniques for unified clinical reasoning. Interest in patient-specific and adaptive monitoring has increased in recent years as they have proved to be more effective in identifying the potential health risks and specific clinical symptoms of an individual, compared to the conventional population-based generic diagnostic flows [254] – [258].

Multi-parameter medical monitoring [259] and multi-sensor data fusion [260] are popular techniques for unified clinical reasoning and improve the robustness of a system by exploiting inherent redundancy in sensor data and signal processing. These techniques are particularly useful for monitoring in extreme circumstances and critical environments where the analysis of intrinsically correlated signals is required, such as intensive care units [261], battlefields [262], and outer space [263]. There are a variety of related works that use multi-parameter monitoring [264] along with data aggregation and fusion [265], [266] but the majority of them focus on analysis of individual measurements or use of correlation between the measured signals to reduce false alarms generated from the monitors. A few related works used multi-parameter physiological signals such as ECG and EEG to compile new measures for prediction or detection of critical conditions such as epileptic seizures or cognitive decline [267], [268].

C.3. Overall Approach

We propose a novel monitoring flow and reconfigurable architecture for real-time monitoring of patient's health status to address the aforementioned challenges. The proposed approach comprises the following parts:

1. **Multi-parameter signal analysis and data fusion** that allows real-time masking of errors by simultaneous monitoring of multiple physiological signals, extracting features of these signals, and synergistic combination of multiple features to compile a personalized health index that can accurately characterize an individual’s health status.

2. **Dynamic reconfiguration supported in hardware** that integrates heterogeneous computing modules through novel reconfiguration strategies to meet changing requirements of the application, performance, and reliability in energy- and cost-effective ways.
3. **Algorithm- and application-specific low overhead detection techniques** that enable real-time detection and diagnosis of errors in the computing engines and recovery from failures while considering power and performance constraints.

Section C.3.1, Section C.3.2, and Section C.3.3 describe each part in more detail.

C.3.1. Multi-Parameter Signal Analysis and Data Fusion

Figure C.1 shows our proposed patient-specific multi-parameter monitoring flow which is based on aggregation of the features and analysis results from inter-correlated physiological signals. The monitoring flow starts with an initial training phase, in which a physiological signature of the individual (Health Index), is compiled by aggregating (constructing a vector of) different statistical features from the input signals. During the monitoring phase, the constructed signature is used as a reference point (patient-specific threshold) for detecting abnormalities in each signal. At the end, a fusion technique (such as a majority voting process) is employed to reach a final diagnostic decision. The data fusion unit can perform different levels of fusion (spanning from data- to feature- and decision-level fusion) according to specific diagnostic needs or the feedback from the results.

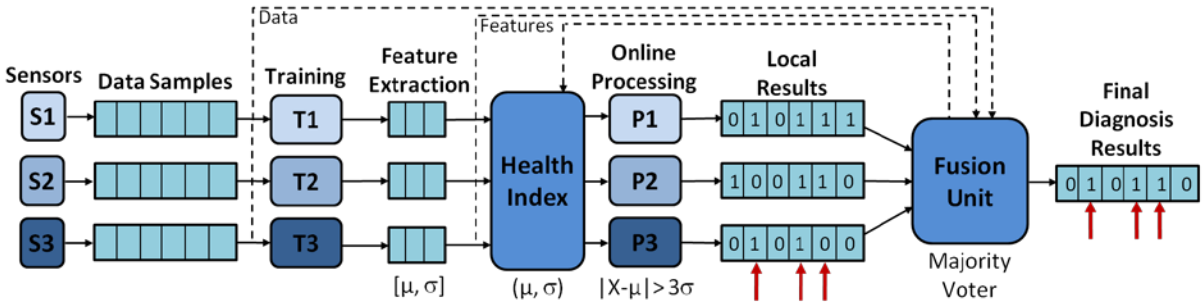


Figure C.1. Patient-specific multi-parameter signal analysis and fusion.

Figure C.2 shows an example from our initial experiments, where we extracted simple statistical features such as mean and standard deviation from the systolic arterial blood pressure (ABP Systolic) and heart rate (HR) signals and used median filtering to compile a template of patient's

ECG waveforms over a training period of 2-hours. The left part shows a 10-minute period of physiological signals along with their corresponding patient-specific thresholds (green lines) computed for a patient (diagnosed with CHF/pulmonary edema), from the publicly available MIMIC database [269]. The right part shows the patient-specific alarms generated based on each individual signal as well as the alarms generated by a majority voter fusion unit (last row). The red bars correspond to the alarms that are masked by the fusion unit.

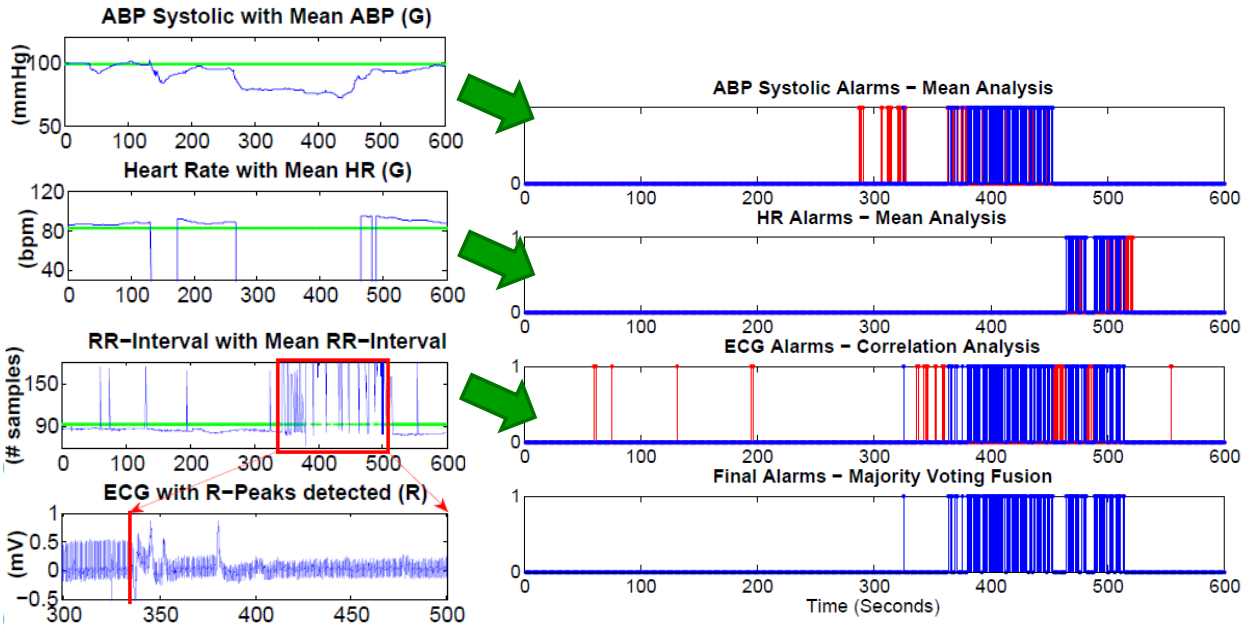
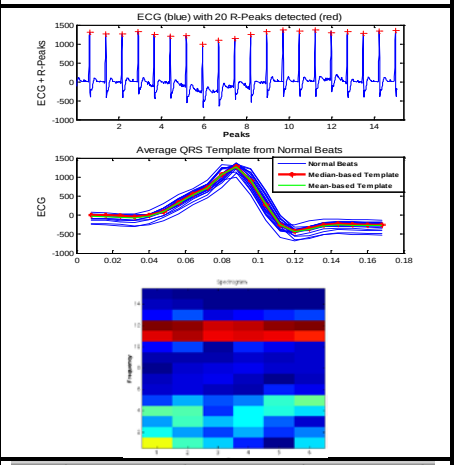
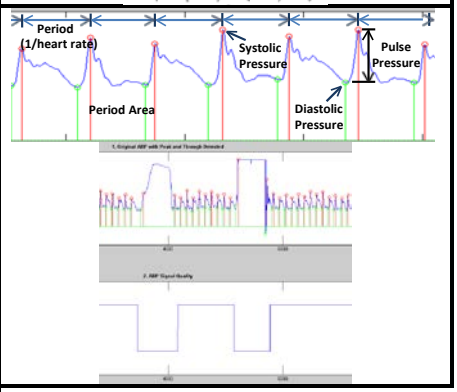


Figure C.2. Patient-specific multi-parameter signal analysis and fusion for a patient from MIMIC database.

Our preliminary analysis of sample periods of up to one hour of patient data collected from the MIMIC database, showed that the decision fusion technique based on simple majority voting is effective in masking the false alarms raised due to patient movements and artifacts [248]. However, in our further experiments on longer periods of patient data (41 hours), we found that around 2% of ICU monitor alarms raised by the ICU monitors were potentially false alarms, because they occurred in a close-proximity of a monitor noise alarm. But the simple majority voting mechanism masked about 98% of the ICU alarms from which only 2% were actually false alarms, i.e. it only had a masking accuracy of 2% [247].

By an intensive study of related works [270] – [275] on detection and prediction of different cardiac events and particularly arrhythmia, we found various features that can be extracted from ECG and ABP waveforms and vital signs. Table C.1 shows the list of features that we implemented in our experiments. We used the time-series ECG, ABP, and vital signs data from MIMIC II multi-parameter database [276], collected from patients in cardiac intensive care units. A subset of the patient data included the expert-reviewed annotations on the patient alarms, which we used as golden alarms in our analysis. Figure C.3 shows six features ($ADB_{variance}$, $ADB_{max\text{-}mean5}$, $ADB_{mean\text{-}top5}$, R-R interval, Mean Energy, and Heart Rate) extracted for every 20-sec window extracted for the patient a40022 (horizontal axes correspond to 20-sec processing windows). The first row shows the golden alarms, indicating the cardiac abnormalities and the red circles highlight the changes in the features in a close proximity of a golden alarm.

Table C.1. Features extracted from the ECG and ABP waveforms.

Signals		Features	Example Features Extracted
Waveform	ECG	• R Peaks • R-R Intervals • ECG Template • Area under each beat (QRS-complex) • ADB: Area difference of beat with mean area of beats(20sec window) • ADB_{mean_top5}, ADB_{max_mean5}, $ADB_{variance}$, $ADB_{gradient}$ • Mean Energy in Lower Frequencies of ECG • Kurtosis of ECG • Heart Rate (20 sec window)	
	ABP	• ABP Peak Detection • BBI: Beat-Beat-Intervals • SBP: Systolic Blood Pressure • DBP: Diastolic Blood Pressure • MBP: Mean Blood Pressure • PP: Pulse Pressure (SBP-DBP) • AREA: Area of the Period • BBI_diff: Difference between two BBI • SBP_diff: Difference between two SBP • DBP_diff: Difference between two BBI • Signal Quality Features (SQI)	
Vital Signs		• Heart Rate, ABPSys, ABPDias, ABPMean, RESP, Spo2	

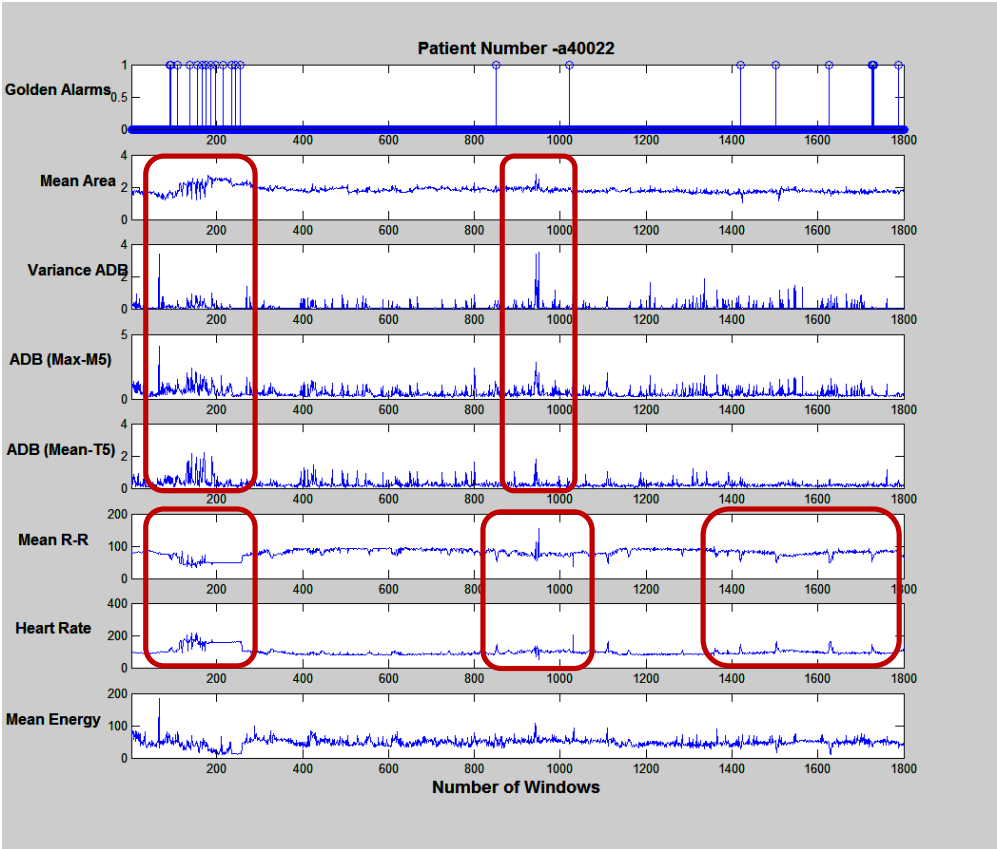


Figure C.3. Golden alarms and extracted features from ECG signal for a patient from MIMIC II database.

We designed two classifiers based on multivariate Gaussian distributions and logistic regressions for detecting the abnormalities using the extracted features. We trained the classifiers based on different periods of training data (100 to 1500 windows) and performed the testing data on the remaining set of samples to label each 20-sec window as an alarm (Class 1) or no-alarm (Class 0) event. Then we compared the results of these classifiers with the golden alarms (as indicated by the expert-reviewed annotations) and computed a confusion matrix showing the number of true/false indications of alarm/no alarm events. We used sensitivity and specificity as the measure to evaluate the accuracy of the results.

Figure C.4 shows the performance of the Gaussian and logistic regression classifiers for the same patient data shown in Figure C.3.

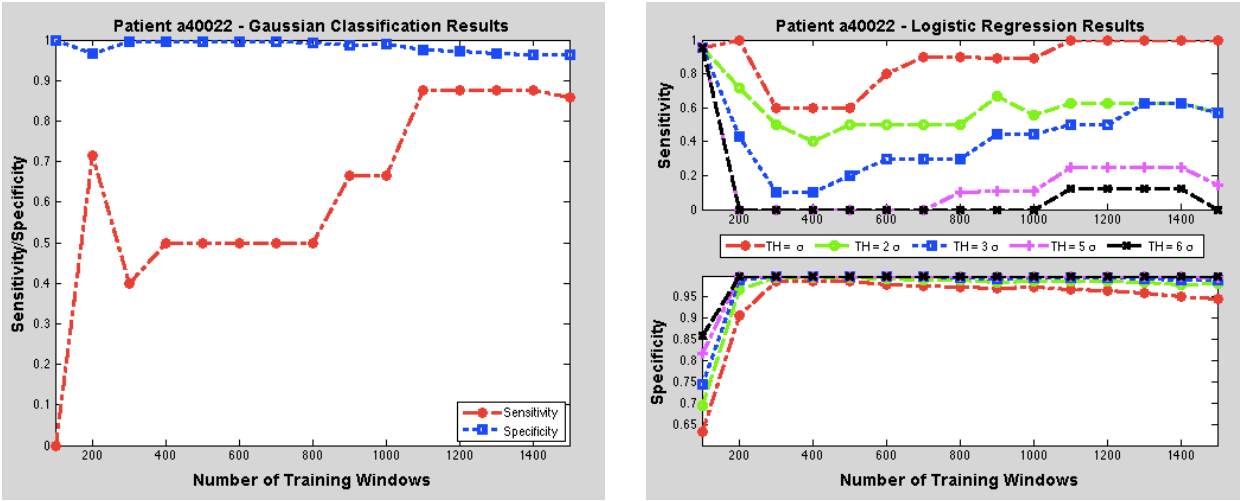


Figure C.4. Sensitivity and specificity of Gaussian and logistic regression techniques in classifying the cardiac events.

The major results from our experiments can be summarized as follows:

- For both the classifiers, an increase in the size of training data improves in the sensitivity of the results. For the Gaussian classifier, a sensitivity of 87% and a specificity of 97% is achieved with training sizes of larger than 1200 windows (> six hours).
- For the logistic regression classifier, with increasing the threshold, the sensitivity tends to decrease and specificity increases.
- Depending on the value of the threshold selected for logistic regression classifier, sensitivity values vary while specificity is almost always high and varies from 63% to 100%. This is because number of “no alarm” cases is higher than “alarm” cases. Among five thresholds, the one-sigma (red) threshold led to the highest sensitivity and specificity for all the tested patients.

C.3.2. Dynamic Reconfigurable Architecture

As shown in Section C.3.1, accurate real-time monitoring of the health status requires concurrent recording and analysis of multiple biomedical signals collected at relatively high sampling rates (125-360 Hz) from the patient's body. Therefore, patient monitors face challenges in real-time parallel processing of large amounts of data under tight power and area constrains. So in order to support adaptive and multi-parameter medical monitoring in real time, an embedded

medical device should exhibit both flexibility and circuit customization. A flexible design can be dynamically reconfigured in the field to meet changing application and performance requirements. Circuit customization on the other hand allows meeting the timing constrains and achieving high throughput, low energy consumption, and small silicon area [277].

We proposed an application-specific reconfigurable architecture for patient-specific multi-parameter medical monitoring with a trade-off between flexibility and circuit customization. As shown in Figure C.5, based on a hybrid hardware/software approach, the proposed architecture incorporates a set of coarse-grained reconfigurable, a configurable communication block, a configurable data fusion unit, and an embedded main processor. The processing elements (PE) are dynamically reconfigurable computing engines, designed by optimized integration of a common set of algorithms for feature extraction from biomedical signals (see Section 3.3). The homogeneous PEs and the flexible communication block, although developed for multi-parameter signal analysis, inherently introduce redundancy in both input data and computational engines, which further enables improved accuracy and reliability, particularly in the face of sensor failures or artifacts in data.

An initial prototype of the proposed architecture was developed on an FPGA platform and was evaluated by running the monitoring flows presented in Section C.3.2, using real patient and device status data collected from the MIMIC database, as well as the algorithms for real-time seizure detection, presented in [278]. Our prototype utilized only around 28% of look-up tables (LUTs) and 17% of slice registers from the whole FPGA resources and other than the embedded processor, the custom logic parts consume only around 300 μW of dynamic power.

For more details on this work can be found in [246], [248], and [249].

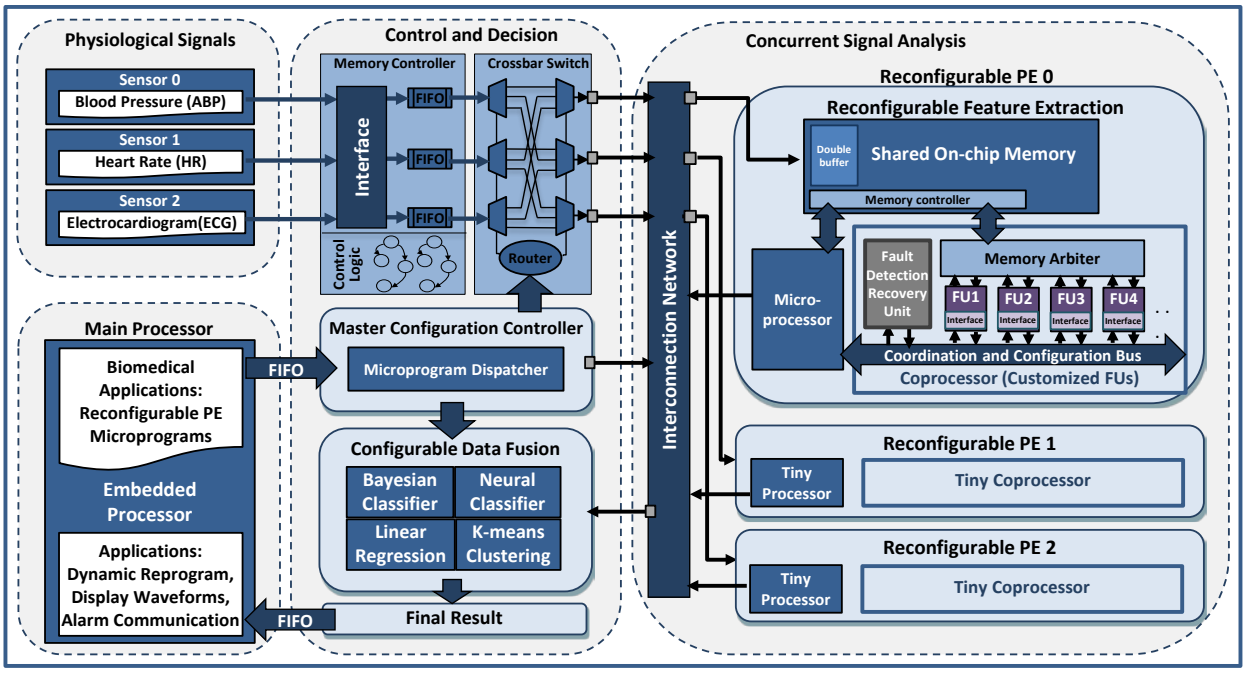


Figure C.5. Reconfigurable architecture for patient-specific multi-parameter monitoring.

C.3.3. Low Overhead Detection and Recovery Techniques

As mentioned in Section C.1, an important safety requirement for patient monitoring devices is to continuously deliver trustworthy results even in the presence of faults to prevent catastrophic impacts on patients. However, along with the scaling of transistor technology, the electrical and hardware components become more susceptible to both transient and permanent faults. However, traditional techniques based on double and triple modular redundancy introduce large area and power overheads and the selective hardware replication techniques cannot provide 100% protection against the faults.

We proposed a runtime reconfigurable feature extraction architecture for the processing elements in the architecture of Figure C.5 that enables trustworthy computation under performance and power constraints. The proposed architecture is composed of a set of functional units that serve as accelerators for commonly used feature extraction operations in biomedical monitoring applications. The functional units are configured and scheduled to run the desired application through extended instruction set of a lightweight embedded processor shown in Figure C.6.

The embedded microprocessor (a lightweight MIPS processor in our prototype) is responsible for: (i) configuring the functional units by sending configuration parameters to them, (ii)

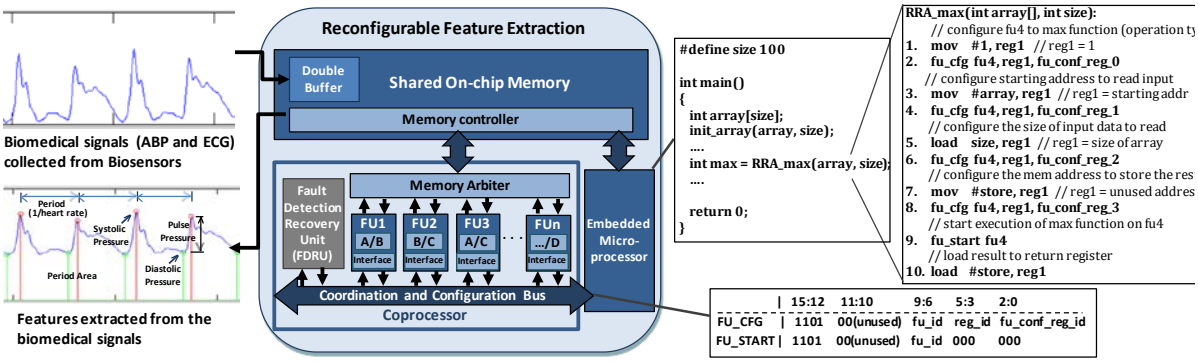


Figure C.6. (a) Input biomedical signals and features extracted by the architecture, b) Architecture overview, (c) Example codes and extended instructions.

scheduling the execution of operations on the functional units by sending the execution instructions, and (iii) executing the basic MIPS instructions that are not supported by the coprocessor (e.g., control flow). The following strategies were employed to enable efficient detection and recovery from computation faults in the functional units with small hardware footprint and low energy consumption:

- Coarse-grained reconfigurable functional units (FUs) designed for the shared computational kernels, which can be programmed within a few cycles to perform different kinds of biomedical signal processing and enable energy-efficient computations in real time. The FUs are designed by following a template with the same interface to allow architecture extension to support other biomedical applications, such as breathing rate monitoring.
- A low-overhead hardware fault detection and recovery unit (FDRU) that monitors the activities of FUs using a configurable watchdog timer and patient-specific invariant checking and upon detection of faults will reset and re-execute the affected FU in real time to dynamically recover from the unexpected faults.

Our initial experiments of running a heart monitoring application (including feature extraction from ABP and ECG signals and heart rate estimation using Kalman filters) on both FPGA and ASIC prototypes of the architecture achieved better performance and energy efficiency compared to an Android implementation of the same algorithm. The proposed architecture could recover from transient faults with low resource (~15%) and energy (~34%) overheads and no (0%) performance impact. More details on this work can be found in [246].

REFERENCES

[1] U.S. Food and Drug Administration, "Medical Device Databases: Recalls, MAUDE, and TPLC" [Online]. Available: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Databases/default.htm.

[2] H. Alemzadeh, R. K. Iyer, Z. Kalbarczyk, J. Raman, "Analysis of safety-critical computer failures in medical devices," *IEEE Security & Privacy*, vol. 11, no. 4, pp. 14-26, July-Aug. 2013, doi:10.1109/MSP.2013.49.

[3] H. Alemzadeh, R. Hoagland, Z. Kalbarczyk, R. K. Iyer, "Automated classification of computer-based medical device recalls," in *Proc. of the 27th IEEE International Symposium on Computer-Based Medical Systems (CBMS'2014)*, 2014, pp. 553-554.

[4] D. Wallace and D. Kuhn, "Failure modes in medical device software: An analysis of 15 years of recall data," *International Journal of Reliability Quality and Safety Engineering*, vol. 8, pp. 351–372, 2001.

[5] B. Zhivko, G. Mitalas, and N. Pallikarakis. "Analysis and classification of medical device recalls," in *World Congress on Medical Physics and Biomedical Engineering*, 2006, pp. 3782-3785.

[6] K. Fu, "Trustworthy medical device software. In public health effectiveness of the FDA 510(k) clearance process: Measuring postmarket performance and other select topics: Workshop report," Institute of Medicine (IOM), National Academies Press, Washington, DC, 2011.

[7] H. Yang and W. Hyman, "An analysis of software-related recalls of medical devices," *Journal of Clinical Engineering*, vol. 35, no. 3, pp. 153, 2010.

[8] W. H. Maisel et al., "Recalls and safety alerts involving pacemakers and implantable cardioverter-defibrillator generators," *JAMA: The Journal of the American Medical Association*, vol. 286, no.7, pp. 793-799, 2001.

[9] D. B. Kramer et al., "Security and privacy qualities of medical devices: An analysis of FDA postmarket surveillance," *PLOS ONE*, vol. 7, no. 7, pp. 1–7, 2012.

[10] International Electrotechnical Commission. *IEC 61025 Fault Tree Analysis*, 1990.

[11] D. H. Stamatis, *Failure Mode and Effect Analysis: FMEA from Theory to Execution*. ASQ Quality Press, 2003.

[12] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2011.

[13] Intuitive Surgical Inc., "The da Vinci® Surgical System" [Online]. Available: http://www.intuitivesurgical.com/products/davinci_surgical_system/.

[14] M. J. Lum et al., “The RAVEN: Design and validation of a telesurgery system,” The *International Journal of Robotics Research*, vol. 28, no. 9, pp. 1183-1197, 2009.

[15] B. Hannaford, et al., “Raven-II: An open platform for surgical robotics research,” *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 4, pp. 954-959, 2013.

[16] H. Alemzadeh, “MedSafe: Automated classification of computer-related recalls” [Online]. Available: https://github.com/homa-alem/MedSafe_Backend.

[17] H. Alemzadeh, “MedSafe: Automated extraction of technical phrases and their relations to create ontology instances” [Online]. Available: https://github.com/homa-alem/MedSafe_Ontology.

[18] U.S. Food and Drug Administration, “Class 2 Device Recall Vivid E9 ultrasound system, Recall number Z-0373-2014, Nov. 21, 2013” [Online]. Available: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm?id=122428.

[19] S. Bird, E. Klein, E. Loper, *Natural Language Processing with Python*, O'Reilly Media, Inc., 2009.

[20] International Organization for Standardization, *ISO 14971 - Medical Devices -- Application of Risk Management to Medical Devices*, 2000.

[21] U.S. Food and Drug Administration, “Guidance for Industry and FDA Staff - Total Product Life Cycle: Infusion Pump - Premarket Notification [510(k)] Submissions,” Apr. 2010 [Online]. Available: http://www.fda.gov/medicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm206153.htm.

[22] D. E. Arney et al., “Generic infusion pump hazard analysis and safety requirements version 1.0,” Department of Computer and Information Science, University of Pennsylvania, Technical Report No. MS-CIS-08-31, Feb. 2009.

[23] U.S. Food and Drug Administration, “Guidance for Industry - Q9 Quality Risk Management,” Jun. 2006 [Online]. Available: http://www.fda.gov/downloads/Drugs/.../Guidances/ucm073511.pdf.

[24] C. A. Ericson, “Event tree analysis,” in *Hazard Analysis Techniques for System Safety*, 2005, pp. 223-234.

[25] J. Dunjó et al. “Hazard and operability (HAZOP) analysis: A literature review.” *Journal of Hazardous Materials*, vol. 173, no. 1, pp. 19-32, 2010.

[26] K. Pattabiraman et al., “SymPLFIED: Symbolic program-level fault Injection and error detection framework,” in *Proc. IEEE International Conference on Dependable Systems and Networks (DSN)*, 2008, pp. 472-481.

[27] K. Pattabiraman, Z. Kalbarczyk, and R. K. Iyer. “Automated derivation of application-aware error detectors using static analysis: The trusted ILLIAC approach.” *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 1, pp. 44-57, 2011.

[28] U.S. Food and Drug Administration, "External defibrillator improvement initiative," Nov. 2010 [Online]. Available: http://www.fda.gov/downloads/MedicalDevices/ProductsandMedicalProcedures/CardiovascularDevices/ExternalDefibrillators/UCM233824.pdf.

[29] U.S. Food and Drug Administration, "Pulse Oximeters - Premarket Notification Submissions [510(k)s] - Guidance for Industry and FDA Staff," Mar. 2013 [Online]. Available: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm341718.htm.

[30] R. Jetley, S. P. Iyer, and P. Jones, "A formal methods approach to medical device review," *IEEE Computer*, vol. 39, no. 4, pp. 61–67, Apr. 2006.

[31] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Vol. 1. Cambridge: Cambridge University Press, 2008.

[32] CX. Zhai and S. Massung. *Text Data Analysis and Management: A Practical Introduction to Text Mining and Information Retrieval.* Association for Computing Machinery and Morgan & Claypool Publishers, 2015, New York, NY.

[33] S. Massung and C. Geigle, "MeTA: Modern Text Analysis" [Online]. Available: https://meta-toolkit.org.

[34] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337–351, 2010.

[35] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer, "Failure data analysis of a LAN of Windows NT based computers," in *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, IEEE, 1999, pp. 178–187.

[36] M. Cinque, D. Cotroneo, Z. Kalbarczyk, and R. Iyer, "How do mobile phones fail? A failure data analysis of Symbian OS smart phones," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (DSN'07), IEEE, 2007, pp. 585–594.

[37] C. Lim, N. Singh, and S. Yajnik, "A log mining approach to failure analysis of enterprise telephony systems," in *IEEE International Conference on Dependable Systems and Networks (DSN 2008)*. IEEE, 2008, pp. 398–403.

[38] S. Matz, L. Votta, and M. Malkawi, "Analysis of failure and recovery rates in a wireless telecommunications system," in *IEEE International Conference on Dependable Systems and Networks (DSN 2002)*. IEEE, 2002, pp. 687–693.

[39] M. Cinque, D. Cotroneo, and S. Russo, "Collecting and analyzing failure data of bluetooth personal area networks," in *IEEE International Conference on Dependable Systems and Networks (DSN 2006)*. IEEE, 2006, pp. 313–322.

[40] F. Magrabi et al., "An analysis of computer-related patient safety incidents to inform the development of a classification," *Journal of the American Medical Informatics Association*, vol. 17, no. 6, 2010, pp. 663-670.

[41] F. Magrabi et al., “Patient safety problems associated with heathcare information technology: An analysis of adverse events reported to the US Food and Drug Administration,” *AMIA Annual Symposium Proceedings*, 2011, pp. 853.

[42] F. Magrabi et al., “Using FDA reports to inform a classification for health information technology safety problems,” *Journal of the American Medical Informatics Association*, vol. 19, no. 1, 2012, pp. 45-53.

[43] F. Castro et al., “An analysis of food and drug administration medical device reports relating to total joint components,” The *Journal of Arthroplasty*, vol. 12, no. 7, pp. 765–771, 1997.

[44] S. Brown et al., “Infusion pump adverse events: Experience from medical device reports,” *Journal of Intravenous Nursing*, vol. 20, no. 1, p. 41, 1997.

[45] J. Brixey et al., “Evaluating a medical error taxonomy,” in *Proceedings of the AMIA Symposium*. American Medical Informatics Association, 2002, p. 71.

[46] R. Hauser and B. Maron, “Lessons from the failure and recall of an implantable cardioverter-defibrillator,” *Circulation*, vol. 112, no. 13, pp. 2040–2042, 2005.

[47] H. Alemzadeh, J. Raman, N. Leveson, R. K. Iyer, “Safety implications of robotic surgery: A study of 13 years of FDA data on da Vinci surgical systems,” presented as J. Maxwell Chamberlain Memorial Paper in Adult Cardiac Surgery at the 50th Annual Meeting of the Society of Thoracic Surgeons (STS), Jan. 2014, CSL Technical Report, UILU-ENG-13-2208.

[48] H. Alemzadeh, R. K. Iyer, Z. T. Kalbarczyk, N. Leveson, J. Raman, “Adverse events in robotic surgery: A retrospective study of 14 years of FDA data,” to appear in *PLOS ONE Journal*, 2016 [Online]. Available: http://arxiv.org/abs/1507.03518.

[49] H. Alemzadeh, J. Raman, N. Leveson, Z. T. Kalbarczyk, R. K. Iyer, “Adverse events in robotic surgery: Systems-theoretic analysis of incident reports,” CSL Technical Report [Online]. Available: http://web.engr.illinois.edu/~alemzad1/papers/Robotic_Surgery_STAMP_Analysis_arxiv.pdf.

[50] H. Alemzadeh, Z. Kalbarczyk, R. K. Iyer., T. Kesavadas, S. Small, J. Raman, “Simulation-based training for safety incidents: Lessons from analysis of adverse events in robotic surgical systems,” presented in the American College of Surgeons' 8th Annual Meeting of the Consortium of ACS-accredited Education Institutes, March 2015.

[51] J. Reason, E. Hollnagel, and J. Paries. “Revisiting the Swiss Cheese model of accidents,” *Journal of Clinical Engineering*, vol. 27, pp. 110-115, 2006.

[52] W. Wu et al., “Effectiveness and efficiency of root cause analysis in medicine,” *Journal of American Medical Association (JAMA)*, vol. 299, no. 6, pp. 685-687, 2008.

[53] C. Nemeth et al., “Afterwords: The quality of medical accident investigations and analyses,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 48, no. 17, pp. 2084-2088, 2004.

[54] C. A. Vincent, "Analysis of clinical incidents: A window on the system not a search for root causes," *Quality and Safety in Health Care*, vol. 13, no. 4, pp. 242-243, 2004.

[55] T. Ishimatsu et al. "Hazard analysis of complex spacecraft using systems-theoretic process analysis," *Journal of Spacecraft Rockets*, vol. 51, no. 2, pp. 509-522, 2014.

[56] V. Balgos, "A systems theoretic application to design for the safety of medical devices," M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 2012.

[57] B. Antoine, "Systems theoretic hazard analysis (STPA) applied to the risk review of complex systems: An example from the medical device industry," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2013.

[58] A. Abdulkhaleq and S. Wagner, "A software safety verification method based on system-theoretic process analysis," in *Computer Safety, Reliability, and Security*, pp. 401-412. Springer International Publishing, 2014.

[59] A. Abdulkhaleq and S. Wagner, "Integrated safety analysis using systems-theoretic process analysis and software model checking," in *Computer Safety, Reliability, and Security*, pp. 121-134. Springer International Publishing, 2015.

[60] M. Ouyang, L. Hong, M. Yu, and Q. Fei, "STAMP-based analysis on the railway accident and accident spreading: Taking the China–Jiaoji railway accident for example," *Safety Science*, vol. 48, no. 5, pp. 544-555, 2010.

[61] T. Song, D. Zhong, and H. Zhong, "A STAMP analysis on the China-Yongwen railway accident," in *Computer Safety, Reliability, and Security*, pp. 376-387. Springer Berlin Heidelberg, 2012.

[62] U.S. Food and Drug Administration, "MAUDE Adverse Event Report: Intuitive Surgical, Inc. Da Vinci S Surgical System Endoscopic Instrument Control System," Jul. 2008 [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=2240665.

[63] "Adverse Event Reporting of Medical Devices," Department of Health and Human Services, Office of Inspector General (OEI-01-08-00110), October 2009 [Online]. Available: https://oig.hhs.gov/oei/reports/oei-01-08-00110.pdf.

[64] U.S. Food and Drug Administration, "MAUDE: Manufacturer and User Facility Device Experience Database," [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/search.CFM.

[65] R. G. Hauser et al., "Deaths and cardiovascular injuries due to device-assisted implantable cardioverter–defibrillator and pacemaker lead extraction," *Europace*, vol. 12, no. 3, pp. 395–401, 2010.

[66] M. Cooper et al., "Underreporting of robotic surgery complications," *Journal for Healthcare Quality*, doi: 10.1111/jhq.12036, 2013.

[67] D. Gildea and D. Jurafsky. "Automatic labeling of semantic roles," *Computational Linguistics*, vol. 28, no. 3, pp. 245-288, 2002.

[68] S. Pradhan, et al., “Shallow semantic parsing using support vector machines,” *HLT-NAACL*, pp. 233-240. 2004.

[69] I. Androutsopoulos and P. Malakasiotis, “A survey of paraphrasing and textual entailment methods,” *Journal of Artificial Intelligence Research*, pp. 135-187, 2010.

[70] D. M. Blei, “Probabilistic topic models,” *Communications of the ACM*, vol. 55, no. 4, pp. 77-84, 2012.

[71] A. Kalyanpur and J. W. Murdock, “Unsupervised entity-relation analysis in IBM Watson,” in *Proceedings of the Third Annual Conference on Advances in Cognitive Systems ACS*, 2015, p. 12.

[72] Intuitive Surgical Inc., “Annual Report,” 2013 [Online]. Available: http://phx.corporate-ir.net/External.File?item=UGFyZW50SUQ9MjIzOTk3fENoaWxkSUQ9LTF8VHlwZT0z&t=1.

[73] MAKO Surgical Corp., “RIO® Robotic Arm Interactive System,” [Online]. Available: http://www.makosurgical.com/physicians/products/rio.

[74] P. Kazanzidesf et al., “An open-source research kit for the da Vinci Surgical System,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6434-6439.

[75] da Vinci Surgery, “da Vinci® Procedures,” [Online]. Available: http://www.davincisurgery.com/da-vinci-surgery/da-vinci-procedures/.

[76] Intuitive Surgical, Inc., “EndoWrist® Instrument & Accessory Catalog,” Jan. 2013 [Online]. Available: http://www.intuitivesurgical.com/products/871145_Instrument_Accessory_%20Catalog.pdf.

[77] Intuitive Surgical, Inc., “Investor Presentation, Q1 2013” [Online]. Available: http://phx.corporateir.net/External.File?item=UGFyZW50SUQ9MTcwOTQ5fENoaWxkSUQ9LTF8VHlwZT0z&t=1.

[78] Intuitive Surgical, Inc., “Intuitive surgical comments on medical device reporting practices,” Mar. 2013 [Online]. Available: http://investor.intuitivesurgical.com/phoenix.zhtml?c=122359&p=irol-newsArticle&id=1796011.

[79] U.S. Food and Drug Administration, “Computer-Assisted (Robotic) Surgical Systems: What are computer-assisted (robotic) surgical systems?,” Nov. 2013 [Online]. Available: http://www.fda.gov/MedicalDevices/ProductsandMedicalProcedures/SurgeryandLifeSupport/ComputerAssistedRoboticSurgicalSystems/default.htm.

[80] U.S. Food and Drug Administration, “Warning Letters,” Jul. 2013 [Online]. Available: http://www.fda.gov/ICECI/EnforcementActions/WarningLetters/2013/ucm363260.htm.

[81] Citron Research, “Has the Halo Been Broken on Intuitive Surgical?,” 2012 [Online]. Available: http://www.citronresearch.com/wp-content/uploads/2012/12/isrg-final1.pdf.

[82] Citron Research, “Intuitive Surgical: Angel with Broken Wings, or the Devil in Disguise?,” Jan. 2013 [Online]. Available: http://www.citronresearch.com/wp-content/uploads/2013/01/Intuitive-Surgical-part-two-final.pdf.

[83] “FDA Investigates Robotic Surgery System after Adverse Event Spike,” Medscape Today [Online]. Available: http://www.medscape.com/viewarticle/803339.

[84] U.S. Food and Drug Administration, “Medical & Radiation Emitting Device Recalls, Class 2 Recall da Vinci S Surgical System IS2000,” Nov. 6, 2013 [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfRES/res.cfm?id=71855.

[85] U.S. Food and Drug Administration, “MAUDE Adverse Event Report, Intuitive Surgical, Inc. Da Vinci S Surgical System Endoscopic Instrument Control System,” [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/Detail.CFM?MDRFOI__ID=1760256.

[86] J. C. Knight and N. G. Leveson, “An experimental evaluation of the assumption of independence in multiversion programming,” *IEEE Transactions on Software Engineering*, no. 1, pp. 96–109, 1986.

[87] J. Bartlett, J. Gray, and B. Horst, “Fault tolerance in Tandem computer systems.” in *The Evolution of Fault-Tolerant Computing*, pp. 55–76. Springer Vienna, 1987.

[88] I. Lee and R. K. Iyer, “Software dependability in the Tandem GUARDIAN system.” *IEEE Transactions on Software Engineering*, vol. 21, no. 5, pp. 455–467, 1995.

[89] V. Ficarra et al., “Retropubic, laparoscopic, and robot-assisted radical prostatectomy: A systematic review and cumulative analysis of comparative studies,” *European Urology*, vol. 55, no. 5, pp. 1037-1063, 2009.

[90] G. Gaia et al., “Robotic-assisted hysterectomy for endometrial cancer compared with traditional laparoscopic and laparotomy approaches: A systematic review,” *Obstetrics & Gynecology*, vol. 116, no. 6, pp. 1422-1431, 2010.

[91] J. F. Boggess et al., “A comparative study of 3 surgical methods for hysterectomy with staging for endometrial cancer: Robotic assistance, laparoscopy, laparotomy,” *American Journal of Obstetrics and Gynecology*, vol. 199, no. 4, pp. 360-e1, 2008.

[92] J. D. Wright et al., “Comparative effectiveness of robotic versus laparoscopic hysterectomy for endometrial cancer,” *Journal of Clinical Oncology*, vol. 30, no. 8, pp. 783-91, 2012.

[93] J. D. Wright et al., “Robotically assisted vs laparoscopic hysterectomy among women with benign gynecologic disease,” *JAMA*, vol. 309, no. 7, pp. 689-698, 2013.

[94] F. Rozet et al., “A direct comparison of robotic assisted versus pure laparoscopic radical prostatectomy: A single institution experience,” *Journal of Urology*, vol. 178, no. 2, pp. 478-482, 2007.

[95] R. Berryhill et al., “Robotic prostatectomy: A review of outcomes compared with laparoscopic and open approaches,” *Urology*, vol. 72, no. 1, pp. 15-23, 2008.

[96] F. Porpiglia et al., “Randomized controlled trial comparing laparoscopic and robot-assisted radical prostatectomy,” *European Urology*, vol. 63, no. 4, pp. 606-614, 2013.

[97] C. Robertson et al., “Relative effectiveness of robot-assisted and standard laparoscopic prostatectomy as alternatives to open radical prostatectomy for treatment of localized prostate cancer: A systematic review and mixed treatment comparison meta-analysis,” *BJU International*, vol. 112, no. 6, pp. 798-812, 2013.

[98] A. L. Rawlings et al., “Robotic versus laparoscopic colectomy,” *Surgical Endoscopy*, vol. 21, no. 10, pp. 1701-1708, 2007.

[99] B. P. Müller-Stich et al., “Robot-assisted versus conventional laparoscopic fundoplication: Short-term outcome of a pilot randomized controlled trial,” *Surgical Endoscopy*, vol. 21, no. 10, pp. 1800-1805, 2007.

[100] S. Breitenstein et al., “Robotic-assisted versus laparoscopic cholecystectomy: Outcome and cost analyses of a case-matched control study,” *Annals of Surgery*, vol. 247, no. 6, pp. 987-993, 2008.

[101] P. K. Edelson et al., “Robotic vs. conventional laparoscopic gastric banding: A comparison of 407 cases,” *Surgical Endoscopy*, vol. 25, no. 5, pp. 1402-1408, 2011.

[102] T. Mihaljevic et al., “Robotic repair of posterior mitral valve prolapse versus conventional approaches: Potential realized,” *The Journal of Thoracic and Cardiovascular Surgery*, vol. 141, no. 1, pp. 72-80, 2011.

[103] S. J. Swanson et al., “Comparing robot-assisted thoracic surgical lobectomy with conventional video-assisted thoracic surgical lobectomy and wedge resection: Results from a multihospital database (Premier),” *The Journal of Thoracic and Cardiovascular Surgery*, vol. 147, no. 3, pp. 929-937, 2014.

[104] M. Kent et al., “Open, video-assisted thoracic surgery, and robotic lobectomy: Review of a national database,” *The Annals of Thoracic Surgery*, vol. 97, no. 1, pp. 236-244, 2014.

[105] J. Lee et al., “Comparison of endoscopic and robotic thyroidectomy,” *Annals of Surgical Oncology*, vol. 18, no. 5, pp. 1439-1446, 2011.

[106] H. Yoo et al., “Comparison of surgical outcomes between endoscopic and robotic thyroidectomy” *Journal of Surgical Oncology*, vol. 105, no. 7, pp. 705-708, 2012.

[107] S. K. Lim et al., “Current status of robot-assisted laparoscopic radical prostatectomy: How does it compare with other surgical approaches?,” *International Journal of Urology*, vol. 20, no. 3, pp. 271-284, 2013.

[108] L. Eichel et al., “Robotics in urologic surgery: Risks and benefits,” *AUA Update Series*, vol. 24, no. 13, pp. 106–111, 2005.

[109] P. M. Kozlowski et al., “Mechanical failure rate of da Vinci robotic system: Implications for pre-op patient counseling” *Journal of Urology*, vol. 175, no. 4, pp. 372–373, 2006.

[110] L. S. Borden et al., “Mechanical failure rate of da Vinci robotic system,” *The Canadian Journal of Urology*, vol. 14, no. 2, pp. 3499-3501, 2007.

[111] K. C. Zorn et al., “Da Vinci robot error and failure rates: Single institution experience on a single three-arm robot unit of more than 700 consecutive robot-assisted laparoscopic radical prostatectomies,” *Journal of Endourology*, vol. 21, no. 11, pp. 1341-1344, 2007.

[112] B. Fischer et al., “Complications of robotic assisted radical prostatectomy,” *World Journal of Urology*, vol. 26, no. 6, pp. 595-602, 2008.

[113] H. J. Lavery et al., “Robotic equipment malfunction during robotic prostatectomy: A multi-institutional study,” *Journal of Endourology*, vol. 22, no. 9, pp. 2165-2168, 2008.

[114] W. S. Ham et al., “Malfunction of da Vinci robotic system—Disassembled surgeon's console hand piece: Case report and review of the literature,” *Urology*, vol. 73, no. 1, pp. 209-e7, 2009.

[115] W. T. Kim, et al., “Failure and malfunction of da Vinci Surgical systems during various robotic surgeries: Experience from six departments at a single institute,” *Urology*, vol. 74, no. 6, pp. 1234-1237, 2009.

[116] D. Kaushik et al., “Malfunction of the da Vinci robotic system during robot-assisted laparoscopic prostatectomy: An international survey," *Journal of Endourology*, vol. 24, no. 4, pp. 571-575, 2010.

[117] M. A. Finan, “Overcoming technical challenges with robotic surgery in gynecologic oncology,” *Surgical Endoscopy*, vol. 24, no. 6, pp. 1256-1260, 2010.

[118] A. C. Mues et al., “Robotic instrument insulation failure: Initial report of a potential source of patient injury,” *Urology*, vol. 77, no. 1, pp. 104-107, 2011.

[119] O. Agcaoglu et al., “Malfunction and failure of robotic systems during general surgical procedures,” *Surgical Endoscopy*, vol. 26, no. 12, pp. 3580-3583, 2012.

[120] C.C. Chen et al., “Malfunction of the da Vinci robotic system in urology,” *International Journal of Urology*, vol. 19, no. 8, pp. 736-740, 2012.

[121] N. C. Buchs et al., “Reliability of robotic system during general surgical procedures in a university hospital,” *The American Journal of Surgery*, vol. 207, no. 1, pp. 84-88, 2014.

[122] Murphy, D., B. Challacombe, O. Elhage, and P. Dasgupta. “Complications in robotic urological surgery,” *The Italian Journal of Urology and Nephrology*, vol. 59, no. 2, pp. 191-198, 2007.

[123] A. Fuller et al., “Electrosurgical injuries during robot assisted surgery: Insights from the FDA MAUDE database,” in *Proceedings of SPIE*, pp. 820714-820714, International Society for Optics and Photonics, 2012.

[124] D. Friedman et al., “Instrument failures for the da Vinci surgical system: A Food and Drug Administration MAUDE database study,” *Surgical Endoscopy*, vol. 27, no. 5, pp. 1503-1508, 2013.

[125] S. Andonian et al., “Device failures associated with patient injuries during robot-assisted laparoscopic surgeries: A comprehensive review of FDA MAUDE database,” *Canadian Journal of Urology*, vol. 15, no. 1, p. 3912, 2008.

[126] S. M. Lucas et al., “Global robotic experience and the type of surgical system impact the types of robotic malfunctions and their clinical consequences: An FDA MAUDE review,” *BJU International*, vol. 109, no. 8, pp. 1222–1227, 2012.

[127] P. Gupta et al., “855 Adverse events associated with the da Vinci surgical system as reported in the FDA MAUDE database,” *Journal of Urology*, vol. 189, no. 4, p. e351, 2013.

[128] E. Manoucheri et al., “MAUDE: Analysis of robotic-assisted gynecologic surgery,” *Journal of Minimally Invasive Gynecology*, vol. 21, no. 4, pp. 592-595, 2014.

[129] F. Marusch et al., “Importance of conversion for results obtained with laparoscopic colorectal surgery,” *Diseases of the Colon & Rectum*, vol. 44, no. 2, pp. 207-214, 2001.

[130] R. Gonzalez et al., “Consequences of conversion in laparoscopic colorectal surgery,” *Diseases of the Colon & Rectum*, vol. 49, no. 2, pp. 197-204, 2006.

[131] A. Simorov et al., “Review of surgical robotics user interface: What is the best way to control robotic surgery?,” *Surgical Endoscopy*, vol. 26, no. 8, pp. 2117-2125, 2012.

[132] M. Seco et al., “Systematic review of robotic minimally invasive mitral valve surgery,” *Annals of Cardiothoracic Surgery*, vol. 2, no. 6, pp. 704-716, 2013.

[133] D. M. Holzhey et al., “Learning minimally invasive mitral valve surgery: A cumulative sum sequential probability analysis of 3895 operations from a single high-volume center,” *Circulation*, vol. 128, no. 5, pp. 483-491, 2013.

[134] J. Lee et al., “The learning curve for robotic thyroidectomy: A multicenter study,” *Annals of Surgical Oncology*, vol. 18, no. 1, pp. 226-232, 2011.

[135] G. I. Barbash, S. A. Glied, “New technology and health care costs--The case of robot-assisted surgery,” *New England Journal of Medicine*, vol. 363, no. 8, pp. 701-704, 2010.

[136] R. J. Damiano Jr, “Robotics in cardiac surgery: The Emperor’s new clothes,” *Journal of Thoracic Cardiovascular Surgery*, vol. 134, no. 3, pp. 559-561, 2007.

[137] F. Robicsek, “Robotic cardiac surgery: Time told!” *Journal of Thoracic Cardiovascular Surgery*, vol. 135, no. 2, pp. 243-246, 2008.

[138] P. Modi et al., “Minimally invasive mitral valve surgery: A systematic review and meta-analysis,” *European Journal of Cardiothoracic Surgery*, vol. 34, no. 5, pp. 943-952, 2008.

[139] R. Dhawan et al., “Multivessel beating heart robotic myocardial revascularization increases morbidity and mortality,” *Journal of Thoracic Cardiovascular Surgery*, vol. 143, no. 5, pp. 1056-1061, 2012.

[140] U.S. Food and Drug Administration, “Small Sample Survey - Final Report, Topic: da Vinci Surgical System,” Nov. 2013 [Online]. Available: http://www.fda.gov/downloads/medicaldevices/productsandmedicalprocedures/surgeryandlifesupport/computerassistedroboticsurgicalsystems/ucm374095.pdf.

[141] R. H. Taylor et al., “Medical robotics and computer-integrated surgery,” *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2008. pp. 1199-1222.

[142] H. C. Lin et al., “Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions,” *Computer Aided Surgery*, vol. 11, no. 5, pp. 220-230, 2006.

[143] H. Alemzadeh, D. Chen, Z. Kalbarczyk, R. K. Iyer, X. Li, T. Kesavadas, J. Raman, “A software framework for simulation of safety hazards in robotic surgical systems,” *SIGBED Review*, vol. 12, no. 4, 2015, *Special Issue on Medical Cyber Physical Systems Workshop*.

[144] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, J. Raman, N. Leveson, R. K. Iyer, “Systems-theoretic safety assessment of robotic telesurgical systems,” in *Computer Safety, Reliability, and Security (SAFECOMP)*, LNCS, vol. 9337, pp. 213-227, Springer International Publishing, 2015.

[145] U.S. Food and Drug Administration, “MAUDE Adverse Event Report: Intuitive Surgical, Inc. da Vinci Surgical System Endoscopic Instrument Control System, MDR report 3035720,” Mar. 2013 [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/Detail.CFM?MDRFOI__ID=3035720.

[146] U.S. Food and Drug Administration, “MAUDE Adverse Event Report: Intuitive Surgical, Inc. da Vinci Surgical System Endoscopic Instrument Control System, MDR report 1743065,” May 2010 [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/Detail.CFM?MDRFOI__ID=1743065.

[147] F. Bilotta et al, “Impact and implementation of simulation-based training for safety,” *The Scientific World Journal*, 2013.

[148] International Organization for Standardization, *ISO 26262: Road vehicles -- Functional safety*, ISO, 2011.

[149] B. O’Connor, “NASA software safety guidebook,” *NASA Technical Standard NASA-GB-8719.13*, 2004.

[150] D. Cotroneo, R. Natella, “Fault injection for software certification,” *IEEE Security and Privacy*, vol. 11, no. 4, pp. 38–45, July-Aug. 2013.

[151] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, “Fault injection techniques and tools,” *Computer* vol. 30, no. 4, pp. 75-82, 1997.

[152] J. Arlat et al., “Fault injection for dependability validation: A methodology and some applications,” *IEEE Trans. Software Engineering*, vol. 16, no. 2, pp. 166–182, 1990.

[153] “RAVEN II Simulator,” [Online]. Available: https://github.com/CSLDepend/raven2_sim.

[154] H. H. King et al., “Plugfest 2009: Global interoperability in telerobotics and telemedicine,” in *IEEE International Conference on Robotic Automation (ICRA)*, pp. 1733–1738, 2010.

[155] Simulated Surgical Systems, “Robotic Surgery Simulator (RoSS),” [Online]. Available: http://www. simulatedsurgicals.com/.

[156] Biorobotics Lab, University of Washington, “RAVEN II Source Code,” [Online]. Available: http://astro.ee.washington.edu/raven2docs/.

[157] M. Quigley et al., “ROS: An open-source Robot Operating System,” *ICRA Workshop on Open Source Software,* vol. 3, no. 3.2, p. 5, 2009.

[158] *Arduino microcontroller* [Online]. Available: http://www.arduino.cc/.

[159] “Safety Assessment of RAVEN II Robot,” [Online]. Available: http://web.engr.illinois.edu/~alemzad1/papers/RAVEN.html.

[160] D. Chen et al., “Error behavior comparison of multiple computing systems: A case study using Linux on Pentium, Solaris on SPARC, and AIX on POWER,” in *Proceedings of the 14th IEEE Pacific Rim Intl. Symp. on Dependable Computing, PRDC*, Dec. 2008.

[161] W. Gu et al., “Characterization of Linux kernel behavior under errors,” in *Proc. International Conference on Dependable Systems and Networks*, pp. 459-68, June 22-25, 2003.

[162] A. Faza et al., “Integrated cyber-physical fault injection for reliability analysis of the smart grid,” in *Computer Safety, Reliability, and Security (SAFECOMP)*, LNCS, vol. 6351, pp. 277–290. Springer, Heidelberg, 2010.

[163] C. Di Martino et al., “Analysis and diagnosis of SLA violations in a production SaaS cloud,” in *25th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 178–188, 2014.

[164] J. D. Park et al., “Method of fault injection for medical device based on ISO 26262,” in *18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, pp. 1–2, 2014.

[165] J. J. Majikes et al., “Literature review of testing techniques for medical device software,” in *4th Medical Cyber-Physical Systems Workshop (MCPS 2013)*, 2013.

[166] E. M. Clarke et al., “Model checking and the state explosion problem,” *Tools for Practical Software Verification*. Springer Berlin Heidelberg, pp. 1-30, 2012.

[167] S. S. Krauss, M. Rejzek, and C. Hilbes, “Tool qualification considerations for tools supporting STPA,” *Procedia Engineering*, vol. 128, pp. 15-24, 2015.

[168] M. Rejzek, S. S. Krauss, C. Hilbes, “Safety driven design with UML and STPA: Talk,” *4th MIT STAMP Workshop*, Massachusetts Institute of Technology, 2015.

[169] R. Smith et al., “Comparative analysis of the functionality of simulators of the da Vinci surgical robot,” *Surgical Endoscopy*, vol. 29, no. 4, pp. 972-983, 2015.

[170] J. Bric et al., “Proficiency training on a virtual reality robotic surgical skills curriculum,” *Surgical Endoscopy*, vol. 28, no. 12, pp. 3343-3348, 2014.

[171] A. Huser et al., “Simulated life-threatening emergency during robot-assisted surgery,” *Journal of Endourology*, 2014.

[172] K. Foell et al., “Robotic surgery basic skills training: Evaluation of a pilot multidisciplinary simulation-based curriculum,” *Canadian Urological Association Journal*, vol. 7, no. 11-12, pp. 430-434, 2013.

[173] A. Patel et al., “Can we become better robot surgeons through simulator practice?,” *Surgical Endoscopy*, vol. 28, no. 3, pp. 847-53, 2014.

[174] S. Seixas-Mikelus et al., “Face validation of novel robotic surgical simulator,” *Urology (Gold)*, vol. 76, no. 2, pp. 357-360, 2010.

[175] A. P. Stegemann et al., “Fundamental skills of robotic surgery: A multi-institutional randomized controlled trial for validation of a simulation-based curriculum,” *Urology*, vol. 81, no. 4, pp. 767-774, 2013.

[176] A. Chowriappa et al., “Development and validation of a composite scoring system for robot-assisted surgical training: The robotic skills assessment score,” *Journal of Surgical Research*, vol. 185, no. 2, pp. 561-569, 2013.

[177] H. Alemzadeh, D. Chen, P. M. Cao, A. Lewis, Z. T. Kalbarczyk, R. K. Iyer, “Targeted attacks on the control system of a teleoperated surgical robot,” *CSL Technical Report*, Sep. 2015 [Online]. Available: http://web.engr.illinois.edu/~alemzad1/papers/RAVEN_Attack_15.pdf.

[178] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, R. K. Iyer, “Targeted attacks on teleoperated surgical robots: Dynamic-model based detection and mitigation,” to appear in *Proc. International Conference on Dependable Systems and Networks*, 2016, Preprints [Online]. Available: http://web.engr.illinois.edu/~alemzad1/papers/Surgical_Robots_Attacks_2015.pdf.

[179] K. Zetter, “Hospital Networks Are Leaking Data, Leaving Critical Devices Vulnerable”, Wired Magazine, 2014 [Online]. Available: http://www.wired.com/2014/06/hospital-networks-leaking-data/.

[180] K. Zetter, “It’s Insanely Easy to Hack Hospital Equipment,” Wired Magazine, 2014 [Online]. Available: http://www.wired.com/2014/04/hospital-equipment-vulnerable/.

[181] SecurityScorecard Insights & News, D. Sears, “Healthcare Breach Shines Spotlight on Third Party Security Risks,” 2015 [Online]. Available: http://blog.securityscorecard.com/2015/06/17/healthcare-breach-shines-spotlight-on-third-party-security-risks/.

[182] Business Wire, F. Wayne, “Nomoreclipboard Notice To Individuals Of A Data Security Compromise,” 2015 [Online]. Available: http://www.businesswire.com/news/home/20150610005964/en/NoMoreClipboard-Notice-Individuals-Data-Security-Compromise.

[183] Healthcare Partners, “Notice to Patients on Privacy Incidents,” April 2015 [Online]. Available: http://www.healthcarepartners.com/PatientResources/PrivacyPractices.aspx.

[184] B. Filkins, “SANS-Norse Health Care Cyberthreat Report,” February 2014 [Online]. Available: https://www.sans.org/reading-room/whitepapers/analyst/health-care-cyberthreat-report-widespread-compromises-detected-compliance-nightmare-horizon-34735.

[185] TrapX Labs - A Division of TrapX Security, Inc., “Anatomy of an Attack – MEDJACK,” 2015 [Online]. Available: http://deceive.trapx.com/AOAMEDJACK_210_Landing_Page.html.

[186] J. Rosen and B. Hannaford, “Doc at a distance,” *IEEE Spectrum*, vol. 43, no. 10, pp. 34-39, 2006.

[187] M. Tozal et al., “On secure and resilient telesurgery communications over unreliable networks,” in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2011, pp. 714-719.

[188] G. S. Lee and B. Thuraisingham, “Cyberphysical systems security applied to telesurgical robotics,” *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 225-229, 2012.

[189] T. Bonaci and H. J. Chizeck, "On potential security threats against rescue robotic systems," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2012, pp. 1-2.

[190] T. Bonaci et al., “Experimental analysis of denial-of-service attacks on teleoperated robotic systems,” in *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, 2015, pp. 11-20.

[191] T. Bonaci et al., "To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots," *arXiv:1504.04339*, 2015.

[192] Cisco Systems Inc., “Cisco PCI Solution for Healthcare” [Online]. Available: https://www.cisco.com/cdc_content_elements/flash/ind_sol/healthcare/PCI_for_Healthcare_Solution_Overview_Final.pdf.

[193] DLR Design Team, “Tilo Wüsthofft Industrial Design” [Online]. Available: http://www.tilo-wuesthoff.de/index.php?/-dlr-design-team/.

[194] M. Bishop and M. Dilger, “Checking for race conditions in file accesses,” *Computing Systems*, vol. 9, no. 2, pp. 131–152, 1996.

[195] W., Jinpeng and C. Pu, “TOCTTOU vulnerabilities in UNIX-Style file systems: An anatomical study,” in *Proceedings of the 4th USENIX Conference on File and Storage Technologies (USENIX FAST)*, vol. 5, pp. 12-12, 2005.

[196] Intuitive Surgical, Inc., “da Vinci® Onsite®” [Online]. Available: http://www.intuitivesurgical.com/support/onsite.html.

[197] N. Falliere, L. O. Murchu, E. Chien, “W32. stuxnet dossier,” Symantec Corp, White Paper 2011.

[198] M. Kerrisk, “LD.SO(8) - Linux Programmer's Manual”, May 2015 [Online]. Available: http://man7.org/linux/man-pages/man8/ld.so.8.html.

[199] Blackhat Academy, InfoSec Institute, “Jynx2 Sneak Peek & Analysis,” March 2012 [Online]. Available: http://resources.infosecinstitute.com/jynx2-sneak-peek-analysis/.

[200] “Azazel,” [Online]. Available: https://github.com/chokepoint/azazel.

[201] “The magic of LD_PRELOAD for Userland Rootkits” [Weblog entry]. *FlUxIuS' Blog*. Available: http://fluxius.handgrep.se/2011/10/31/the-magic-of-ld_preload-for-userland-rootkits/.

[202] “The Rootkit Hunter Project,” [Online]. Available: http://rkhunter.sourceforge.net/.

[203] “Locally checks for signs of rootkit,” [Online]. Available: http://www.chkrootkit.org/.

[204] “Unhide-The OpenSource Forensic Tool,” [Online]. Available: http://www.unhide-forensics.info/.

[205] “MoVP 2.4 Analyzing the Jynx rootkit and LD_PRELOAD,” [Online]. Available: http://volatility-labs.blogspot.com/2012/09/movp-24-analyzing-jynx-rootkit-and.html.

[206] Schweitzer Engineering Laboratories Inc., “SEL-3021-1 Serial Encrypting Transceiver Data Sheet” [Online]. Available: https://www.selinc.com/WorkArea/DownloadAsset.aspx?id=2854.

[207] P. P. Tsang and S. W Smith, “YASIR: A low-latency, high-integrity security retrofit for legacy SCADA systems,” in *Proceedings of The IFIP TC 23rd International Information Security Conference*, 2008, pp. 445-459.

[208] K. Coble, W. Wang, B. Chu, and Z. Li, “Secure software attestation for military telesurgical robot systems,” in *Military Communications Conference, 2010-MILCOM*, pp. 965-970, 2010.

[209] C. Castelluccia, A. Francillon, D. Perito, and C. Soriente, “On the difficulty of software-based attestation of embedded devices,” in *Proceedings of the 16th ACM Conference on Computer and communications security*, pp. 400-409, 2009.

[210] “CVE-2015-5123,” [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-5123.

[211] “CVE-2015-0235,” [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0235.

[212] “CVE-2014-4113,” [Online]. Available: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-4113.

[213] “CVE-2014-6271,” [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271.

[214] "CVE-2015-5783," [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-5783.

[215] M. Haghighipanah, Y. Li, M. Miyasaka and B. Hannaford, “Improving Position Precision of a Servo-Controlled Elastic Cable Driven Surgical Robot Using Unscent Kalman Filter,” in *IROS*, 2015.

[216] “ROS Core Components,” [Online]. Available: http://www.ros.org/core%20components/.

[217] J. Huang et al., “ROSRV: Runtime Verification for Robots,” in *the 5th International Conference on Runtime Verification*, Toronto, ON, Canada, 2014, pp. 247-254.

[218] "ROS Technical Overview," [Online]. Available: http://wiki.ros.org/ROS/Technical%20Overview.

[219] J. McClean, C. Stull, C. Farrar, and D. Mascareas, "A preliminary cyber-physical security assessment of the Robot Operating System (ROS)," in *SPIE Defense, Security, and Sensing*, 2013, pp. 874110-874110.

[220] Intuitive Surgical, Inc. "da Vinci OnSite Site Survey Questionnaire" [Online]. Available: https://www.tfaforms.com/128949.

[221] Qualys, Inc., "QUALYS SSL LABS - SSL Server Test" [Online]. Available: https://www.ssllabs.com/ssltest/.

[222] Qualys, Inc., "SSL Report: dvms-dv.davinci-onsite.com" [Online]. Available: http://bit.ly/1hKLHMV.

[223] B. Moeller and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for preventing protocol downgrade attacks," *Internet Engineering Task Force (IETF)*, RFC 7507, Apr. 2015 [Online]. Available: http://tools.ietf.org/html/rfc7507.

[224] Y. Mo et al., "Cyber–physical security of a smart grid infrastructure," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195-209, 2012.

[225] H. Lin et al., "Runtime semantic security analysis to detect and mitigate control-related attacks in power grids," in *IEEE Transactions on Smart Grid*, 2016.

[226] S. Amin et al., "Stealthy deception attacks on water SCADA systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 161-170. ACM, 2010.

[227] A. Cárdenas et al., "Attacks against process control systems: Risk assessment, detection, and response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pp. 355-366, 2011.

[228] K. Koscher et al., "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy (SP)*, pp. 447-462, 2010.

[229] U.S. Food and Drug Administration, "Class 2 Device Recall iLab Ultrasound Imaging System, models 120INS and 240INS," 2008 [Online]. Available: http://www.accessdata.fda.gov/SCRIPTs/cdrh/cfdocs/cfres/res.cfm?id=73287.

[230] U.S. Food and Drug Administration, "MAUDE Adverse Event Report: PHILIPS MEDICAL SYSTEMS XCELERA LLZ (SYSTEM, IMAGING PROCESSING, RADIOLOGICAL)," 2009 [Online]. Available: http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/Detail.CFM?MDRFOI__ID=1568388.

[231] A. De Luca et al., "Collision detection and safe reaction with the DLR-III lightweight manipulator arm," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 1623-1630.

[232] A. De Luca et al., “Sensorless robot collision detection and hybrid force/motion control,” in *Proceedings of the 2005 IEEE International Conference Robotics and Automation, (ICRA)*. IEEE, 2005, pp. 999-1004

[233] T. Matsumoto and K. Kazuhiro, “Collision detection of manipulator based on adaptive control law,” in *Proceedings of 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics,* vol. 1, 2001, pp. 177-182.

[234] J. P. Mendoza, M. V. Manuela, and R. Simmons. “Mobile robot fault detection based on redundant information statistics,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2012.

[235] J. H. Shin et al., “Fault detection and robust fault recovery control for robot manipulators with actuator failures," in *Proceedings of the IEEE International Conference on Robotics and Automation.* vol. 2, IEEE, 1999, pp. 861-866.

[236] A. De Luca and L. Ferrajoli, “Exploiting robot redundancy in collision detection and reaction,” in the *IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS),* IEEE, 2008, pp. 3299-3305.

[237] K. M. Kavi, “Beyond the black box,” *IEEE Spectrum,* vol. 47, no. 8, pp 46-51, 2010.

[238] J. T. Correia et al., “Utilizing data from automotive event data recorders,” *Proceedings of the Canadian Multidisciplinary Road Safety Conference XII,* London Ontario, 2001.

[239] K. Murphy, “Dynamic Bayesian networks: Representation, inference and learning,” PhD dissertation, University of California, Berkeley, 2002.

[240] P. Weber and L. Jouffe, “Reliability modelling with dynamic Bayesian networks,” in *5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS'03), Washington, DC, USA*, pp. 57-62. IFAC, 2003.

[241] X. Wu et al., “A dynamic Bayesian network based approach to safety decision support in tunnel construction,” *Reliability Engineering & System Safety,* vol. 134, pp. 157-168, 2015.

[242] D. Koller et al., “Towards robust automatic traffic scene analysis in real-time,” in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing. Proceedings of the 12th IAPR International Conference on*, vol. 1, pp. 126-131. IEEE, 1994.

[243] K. Liang et al., “Detection and prediction of adverse and anomalous events in medical robots,” In *IAAI*, 2013.

[244] P. Cao et al., “Preemptive intrusion detection,” in *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, p. 21. ACM, 2014.

[245] P. Cao et al., “Preemptive intrusion detection: Theoretical framework and real-world measurements,” in *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, p. 5. ACM, 2015.

[246] Q. Li, H. Alemzadeh, Z. Kalbarczyk, R. K. Iyer, “A fault-tolerant hardware architecture for robust wearable heart rate monitoring,” in *Proc. of the 9th International Conference on*

Pervasive Computing Technologies for Healthcare (Pervasive Health'2015), 2015, pp. 185-192.

[247] H. Alemzadeh, C. D. Martino, Z. Jin, Z. Kalbarczyk, R. K. Iyer, "Towards resiliency in embedded medical monitoring devices," in *Open Resilient Human-Aware Cyber-Physical Systems, Dependable Systems and Networks Workshops (DSN-W)*, 2012, pp. 1-6.

[248] H. Alemzadeh, Z. Jin, Z. Kalbarczyk, R. K. Iyer, "An embedded reconfigurable architecture for patient-specific multi-parameter medical monitoring," in *33rd Annual International IEEE EMBS Conference (EMBC'11)*, pp. 1896-1900, Sep. 2011.

[249] H. Alemzadeh, M. U. Saleheen, Z. Jin, Z. Kalbarczyk, R. K. Iyer, "RMED: A reconfigurable architecture for embedded medical monitoring," in *Fifth Annual IEEE-NIH Life Science Systems and Application Workshop (LiSSA'11)*, 2011, pp. 112-115.

[250] H. Lin, H. Alemzadeh, D. Chen, Z. T. Kalbarczyk, R. K. Iyer, "Safety-critical cyber-physical attacks: Analysis, detection, and mitigation," to appear in *Proceedings of the Symposium and Bootcamp on the Science of Security (HOTSOS)*, 2016.

[251] G. Neubauer and H. Friedl, "Modelling sample sizes of frequencies," in *Proceedings of the 21st International Workshop on Statistical Modelling*, 3-7 July 2006, Galway, Ireland.

[252] G. Neubauer et al., "Models for underreporting: A Bernoulli sampling approach for reported counts," *Austrian Journal of Statistics*, vol. 40, no. 1 & 2, pp. 85–92, 2011.

[253] D. Machin et al., *Sample Size Tables for Clinical Studies*. BMJ Books, 2008.

[254] Y. Hu et al., "A patient-adaptable ECG beat classifier using a mixture of experts approach," *IEEE Trans. Biomed. Eng.*, vol. 44, no. 9, pp. 891–900, Sep. 1997.

[255] S. Kumar et al., "Ubiquitous computing for remote cardiac patient monitoring: A survey," *Int'l J. Telemed. and Appl.*, vol. 2008, pp. 1–19, 2008.

[256] H. Qu and J. Gotman, "A patient-specific algorithm for the detection of seizure onset in long-term EEG monitoring: Possible use as a warning device," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 2, pp. 115–122, 1997.

[257] A. Shoeb et al., "Impact of patient-specificity on seizure onset detection performance," in *Proc. EMBC*, Aug. 2007, pp. 4110–114.

[258] R. Watrous and G. Towell, "A patient-adaptive neural network ECG patient monitoring algorithm," in *Proc. Computers in Cardiology*, pp. 229–232, Sep. 1995.

[259] U. Anliker et al., "AMON: A wearable multiparameter medical monitoring and alert system," *IEEE Trans. Inf. Technol. Biomed.*, vol. 8, no. 4, pp. 415–427, Dec. 2004.

[260] E. Kenneth et al., "Data fusion of multimodal cardiovascular signals," in *Proc. EMBC*, pp. 4689–4692, Sep. 2005.

[261] L. Tarassenko et al., "BIOSIGN: Multi-parameter monitoring for early warning of patient deterioration," in *Proc. MASP*, 2005, pp. 71–76.

[262] NATO, “Real-time physiological and psycho-physiological status monitoring,” North Atlantic Treaty Organisation,” TR-HFM-132, Jul. 2010.

[263] C. Mundt et al., “A multiparameter wearable physiologic monitoring system for space and terrestrial applications,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 9, no. 3, pp. 382–391, Sep. 2005.

[264] G. Clifford et al., “Robust parameter extraction for decision support using multimodal intensive care data,” *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1887, pp. 411–42, 2009.

[265] N. Kannathal et al., “Cardiac health diagnosis using data fusion of cardiovascular and haemodynamic signals,” *Computer Methods and Programs in Biomedicine*, vol. 82, no. 2, pp. 87–96, May 2006.

[266] L. Thoraval et al., “Data fusion of electrophysiological and haemodynamic signals for ventricular rhythm tracking,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 16, no. 6, pp. 48–55, 1997.

[267] T. Bermudez, D. Lowe, and A.-M. Arlaud-Lamborelle, “EEG/ECG information fusion for epileptic event detection,” in *Proceedings of International Conference on Digital Signal Processing (ICDSP)*, pp. 1-8, July 2009.

[268] B. R. Greene et al., “Combination of EEG and ECG for improved automatic neonatal seizure detection,” *Clinical Neurophysiology*, vol. 118, no. 6, pp. 1348-1359, June 2007.

[269] G. B. Moody and R. G. Mark, “A database to support development and evaluation of intelligent intensive care monitoring,” *Computers in Cardiology*, 1996, pp. 657–660.

[270] J. Pan and W. J. Tompkins, “A real-time QRS detection algorithm,” *IEEE Transactions on Biomedical Engineering*, vol. 3, pp. 230-236, 1985.

[271] Q. Li and G.D. Clifford, “Signal quality and data fusion for false alarm reduction in the intensive care unit,” *Journal of Electrocardiology*, vol. 45, no. 6, pp. 596-603. 2012.

[272] Q. Li, R.G. Mark, G.D. Clifford, “Robust heart rate estimation from multiple asynchronous noisy sources using signal quality indices and a Kalman filter,” *Physiological Measurement*, vol. 29, no. 1, p. 15, 2007.

[273] A. Aboukhalil, L. Nielsen, M. Saeed, R. G. Mark, G. D. Clifford, “Reducing false alarm rates for critical arrhythmias using the arterial blood pressure waveform,” *Journal of Biomedical Informatics*, vol. 41, no. 3, pp. 442–451, 2008.

[274] X. Sun, AT Reisner, RG Mark, “A signal abnormality index for arterial blood pressure waveforms,” *Computers in Cardiology*, 2006, pp. 13-16.

[275] W. Zong, G. B. Moody, and R. G. Mark, “Reduction of false arterial blood pressure alarms using signal quality assessment and relationships between the electrocardiogram and arterial blood pressure,” *Medical and Biological Engineering and Computing*, vol. 42, no. 5, pp. 698–706, 2004.

[276] M. Saeed et al., "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A public-access intensive care unit database." *Critical Care Medicine*, vol. 39, no. 5, p. 952, 2011.

[277] S. Vassiliadis and D. Soudris, *Fine- and Coarse-Grain Reconfigurable Computing*. Netherlands: Springer, 2007.

[278] M. U. Saleheen, H. Alemzadeh, A. M. Cheriyan, Z. T. Kalbarczyk, R. K. Iyer, “An efficient embedded hardware for high accuracy detection of epileptic seizures,” in *Proceedings of 3rd International Conference on BioMedical Engineering and Informatics (BMEI'10)*, 2010.