

Literature Review of Testing Techniques for Medical Device Software

John J. Majikes
Department of Computer
Science
North Carolina State
University
Raleigh, NC, USA
jjmajike@ncsu.edu

Rahul Pandita
Department of Computer
Science
North Carolina State
University
Raleigh, NC, USA
rpandit@ncsu.edu

Tao Xie
Department of Computer
Science
North Carolina State
University
Raleigh, NC, USA
txie@ncsu.edu

ABSTRACT

As software-controlled medical devices evolve from monolithic devices to modular Medical Cyber-Physical Systems (MCPS), the software controls become more complex. An undesirable byproduct of increased complexity is added likelihood for failures in MCPS. Software testing has been used as a means to ensure high-quality software, and is also applicable to the software components of these MCPS. In this paper, we review existing software-testing techniques that expose failures in software-controlled medical devices. In particular, we categorize the failures in existing software-controlled medical devices. Then, we use the Systematic Literature Review (SLR) method to compare existing research studies against the categorized failures. Finally, we suggest improvements that can help future research studies of complex medical devices such as MCPS.

Categories and Subject Descriptors

J.3 [Computer Applications]: Life and Medical Sciences; D.2.5 [Software Engineering]: Testing and Debugging; D.2.4 [Software / Program Verification]: Reliability, Validation

General Terms

Verification

Keywords

Medical Device, Medical Cyber-Physical Systems, Systematic Literature Review, Testing and Debugging

1. INTRODUCTION

Software use in general consumer products has been doubling every two or three years [9]. One subcategory of these consumer products is medical devices. Since medical devices directly deal with public health, safety of these devices is highly critical. Thus, these medical devices are often regulated by agencies to ensure safety. For instance, in the United States of America, the Food and Drug Administration (FDA) regulates medical devices.

However, FDA is often forced to depend on individuals and manufacturers to report medical-device failures¹ that *reasonably suggest that a device may have caused or contributed to a death or serious injury* [1]. Given such dependency (and other considerations such as the device manufacturer's proprietary information), the causes of failures in medical devices are often not publicly available.

In the absence of such public information, how can researchers learn from these reported failures found in FDA Class I recalls? Do testing-research studies expose the reported types of these failures? What kinds of testing-research studies help regulators and developers improve software-controlled medical devices by exposing the types of failures found in FDA Class I recalls? Can research studies on more complicated Medical Cyber-Physical Systems (MCPS) better expose these types of failures?

In order to address these questions, we use the method of Systematic Literature Review (SLR) [26] to compare existing testing-research studies and the types of the reported failures described in the FDA Class I recalls. The SLR also addresses emerging challenges for researchers, regulators, and developers as software controls are increasing pervasive with MCPS.

In the rest of this paper, Section 2 presents the most serious failures, FDA Class I recalls, from years 2001 through 2013 and grounded theory [15] to determine the types of medical devices with reported failures. Section 3 describes the SLR method used to query research literature for investigating failures related to medical devices. Sections 4-7 cover the types of research studies found with the SLR, the studied advanced software-testing techniques that are an enhancement to basic testing, the ability of the techniques to be applied to the FDA Class I recalls, and finally the mapping of the techniques for each device type to such device type's recalls. Section 8 describes how the work of researchers, regulators, and developers can improve future software-controlled medical devices like MCPS.

2. FDA CLASS I RECALLS

In order to determine the effectiveness of research studies at detecting faults that cause failures in software-controlled medical devices, we need to identify and categorize these failures. We take an approach of grounded theory by categorizing the failures of FDA Class I recalls, from the years 2001 through 2013 [13]. The

¹These reported failures are observable errors, which are incorrect internal states caused by executing a fault in the software. Software-testing techniques intend to cover all code portions including the code portion containing a fault (if any exists), cause errors, and propagate these errors to expose them as failures.

Table 2: Search Terms

Testing Terms	Acceptance Testing	Automated Verification System	Automatic Testing	Black Box Testing	Branch Testing
	Component Test	Conformance Test	Concolic Test	Cyber-Physical	Dependability
	Formal Testing	Functional Testing	Glass Box Testing	Informal Testing	Integration Testing
	Model Checking	Model-Driven Development	Module Testing	Mutation Testing	Operational Testing
	Path Testing	Random Testing	Regression Test	Security Testing	Software Testing
	Statement Testing	Static Analysis	Stress Testing	Structural Testing	Symbolic Execution
	System Testing	Test Automation	Testing	Unit Testing	Verification
Medical Device Terms	Anesthesia	Bypass	Central apnea	Circulatory pump	Cortical stimulator
	Compounder	Defibrillator	Diabetic	ECG	FDA
	Glucose meter	Hemodialysis	Infusion pump	Medical imaging	Magnetic resonance
	Medical device	MRI	Pacemaker	Resuscitator	Stimulator
	Surgery	Surgical	Vasodilator	Ventilator	X-ray
Exclusion Terms	AIPmut mutation	AIX mutation	Artery bypass	Automated test equipment	Bypass compiler
	Cardiopulmonary bypass	COX regression	EGFR mutation	Egger's regression	FBN1 mutation
	KRAS mutation	Liver function	Logistic regression	Lung function	Pacemaker cell
	Pancreatic function	Stress testing of ECG devices	Stress test	Secondary mutation	To bypass
	Verification of results				

Table 1: Medical-Device Type and FDA Failure Category

Device Type	Non-Software	General Software	Software Alarm	Medication Calibration	User-Interface	Mechanical Electrical	Total Recalls ²
Anesthesia	0	1	0	0	0	2	2
Central apnea	0	0	1	0	0	0	1
Circulatory Pump	0	0	0	0	0	1	1
Compounder	0	0	1	0	0	0	1
Cortical stimulator	0	1	0	0	0	1	1
Defibrillator	0	12	7	0	1	13	22
General Software	0	4	0	0	1	3	6
Glucose Meter	0	1	0	2	3	1	5
Hemodialysis	0	2	3	0	1	0	3
Infusion pump	2	13	11	8	8	8	31
Non-Software	187	0	0	0	0	0	187
Vasodilator	0	0	0	0	0	1	1
Surgical Tool	0	2	0	0	1	1	3
Ventilator	1	3	4	1	0	8	14
Workstation	0	4	0	1	0	0	4
Total	190	45	27	12	15	39	282

used failure categories are from a 2010 FDA initiative that partitions recall failures into six categories: (1) non-software failures, (2) general software failures, (3) alarm failures, (4) user-interface issues, (5) medication dose calibration, and (6) mechanical or electrical failures [12]. Table 1 shows our categorization of failures and device types. Column “Device Type” lists the 15 device types identified by FDA in the recalls. The remaining columns list the number of recalls in each of the categories (column names) identified by FDA. The detailed results of our work are available at <http://research.csc.ncsu.edu/ase/projects/MedDevTestingSLR>.

3. SLR PROCESS

This section describes the formal SLR process. The four research questions that we investigate are described below:

- RQ1: What types of software-controlled medical devices were investigated in the studies? In addition, future researchers would want to know whether an actual device or some type of simulation was used. The audience that the research studies are targeted at may also be a concern for decisions of future research.

²Failures are classified into multiple categories.

- RQ1a: Did researchers actually test actual medical device software or simulate a medical device?
- RQ1b: Did researchers target a medical or a software engineering audience?
- RQ2: What types of advanced software-testing techniques were investigated in the studies? In addition, future researchers would want to know the techniques used for each medical-device type:
 - RQ2a: Did the types of testing techniques differ across medical-devices types?
- RQ3: How did advanced software-testing techniques improve software-controlled medical devices? To address this general question, we answered the following three questions:
 - RQ3.1: Did recalls provide sufficient information to help identify causes of the failures?
 - RQ3.2: Could advanced software-testing techniques detect faults that cause these types of failures?
 - RQ3.3: Which advanced software-testing techniques could be potentially used by regulators (i.e., without requiring access to source code of the software under test)?
- RQ4: For each type of medical devices, how much overlap exists between the types of failures identified in the studies and the commonly occurring types of device failures?

3.1 SELECTION OF DIGITAL LIBRARIES AND RESEARCH STUDIES

Testing Terms.

Table 2 (Row 1) lists the software-testing terms used in this SLR to query digital libraries for finding research studies that apply advanced testing to detect faults in modern software. These terms have been derived from the IEEE Standard Glossary of Software Engineering Terminology [7] are further refined through literature searches.

Medical Device Terms.

Table 2 (Row 2) lists the medical terms used in this SLR to query digital libraries for finding medical research studies. These terms have been derived by generalizing medical devices found in FDA

Table 3: Digital Library Search Results

Digital Library	Returned Papers	Reduction by Title	Final Selection
ACM Digital Library	50	34	12
Compendex Engineering Library	806	67	24
IEEE Explore Library	288	29	12
MedLine	256	4	1
Web of Science	282	9	5
Total ³	1,682	75	31

Class I recalls listed in Table 1, and are further refined through literature searches.

Exclusion List.

There exist a lot of software-testing technique terms that can be used as keywords to search for research studies pertaining to software testing. However, we have deliberately excluded basic testing terms such as *unit test*, *static analysis*, and *function test*, because we target the advanced testing techniques that are an enhancement to basic testing. Furthermore, certain generic testing terms such as *validation* and *mutation testing* have also been excluded in the query. Such terms have a confounding meaning in the context of medical literature such as *validation of data* and *AIPmut mutation testing*. Inclusion of such terms in the query would flood the results with non-applicable research papers. Table 2 (Row 3) lists the terms that have been excluded from the digital library queries.

After conferring with library research specialists at North Carolina State University, we searched the following electronic databases: ACM Digital Library[®], Compendex[®], IEEE Xplore[®], MedLine[®], and Web Of Science[®]. We searched for research studies from the aforementioned digital libraries using keywords listed in Table 2. Specifically, we formulated the query as $(T \cap M - E)$, where (1) T , the set of research studies containing the testing terms, (2) M , the set of research studies containing the medical terms, and (3) E , the set of research studies containing the exclusion terms.

Table 3 lists the number of research studies (Column ‘Returned Papers’) returned using the query for each library (Column ‘Digital Library’). In all, the queries returned 1,682 research papers. The papers were then reviewed and discussed by the first two authors of this paper to eliminate studies that were unrelated to testing or software-controlled medical devices, or were duplicates across the libraries to produce a set of 31 papers. The detailed results of our categorization are available at <http://research.csc.ncsu.edu/ase/projects/MedDevTestingSLR>.

3.2 STUDY-QUALITY ASSESSMENT

Determining the quality of the research-study selection has a large potential for bias. Therefore, we simply provide information for the readers to judge.

Listing the institutions affiliated with each research study shows a vast collection of universities, government agencies, and corporations from around the world. Note that the FDA and the University of Pennsylvania appeared in 7 papers’ affiliations. Kansas State, Naval Postgraduate School, North Carolina State University, University of Lorraine, and University of Maryland appeared in 2 papers’ affiliations each. In all, we found 24 unique universities, 12 industrial establishments, along with the 2 government agencies appearing in paper affiliations of the studies considered for this SLR.

Listing the publication locations also gives a view of the quality and diversity of publications. 19 of the selected research-studies

³Duplicates are removed.

Table 4: Medical Device Studies

Device name	Research Studies	
	[Actual device]	[Simulated device]
Infusion pump	Kim [24] Lee [28] Jetley [21] Ray1 [36] Ray1 [37] Sankaranarayanan [38] Zafar [42]	
Medical Imaging	Jomier [23] Near [34] Jaring [18] Sayre [39]	
Pacemaker / ICD	Albarghouthi [2] Gomes [16] Jee1 [19] Jee2 [20] Méry1 [30] Méry2 [31] Pajic [35] Ammar [4] David [8] Jiang [22]	
Platform	King [25] Sloane [40] Taneja [41]	
Resuscitator & Infusion Pump	Alur [3] Auguston [6] Drusinsky [10]	
Surgical Tool	Anier [5] Muradore [33] Li [29]	
Ventilator	Miller [32]	

are from conferences, 11 from journals or symposiums, and 1 is from a workshop.

Furthermore, examining the collaborations between different types of institutions (universities, government agencies, and corporations) also gives some indication of the resources used in the study. Walter Reed Army Institute of Research (WRAIR) collaborated with the University of Pennsylvania and the New Jersey Institution of Technology [3]. West Virginia University collaborated with Hewlett-Packard Laboratory and Motorola Labs [4]. VTT Technical Center of Finland collaborated with Vrije University Amsterdam and Intuit [18]. The FDA collaborated with North Carolina State University [21, 41], University of Pennsylvania [24], University of Maryland [36, 37], and Wisdom Software [39]. Kitware Inc. collaborated with Robarts Research Institute and Georgetown University [23].

4. MEDICAL-DEVICE TYPES

The first question that we investigate in this SLR is what types of software-controlled medical devices were investigated in the studies. Device names were collected from the medical devices with failures identified by the FDA and the set of 31 selected research studies. Column 1 (‘Device Name’) of Table 4 lists the device names used in the 31 selected research studies. Column 2 (‘Research Studies’) lists the corresponding research studies along with the first author name. Three of the categorized device names are described below.

The *Medical Imaging* term is a category that includes four studies: (1) an image guided surgical tracking system [23] and (2) a proton therapy center [34], (3) a Magnetic Resonance Imaging (MRI) scanner [18], and (4) a radiation therapy planning system [39].

The *Platform* term is a category that includes three networked medical devices: (1) a Medical Device Plug and Plan (MDPnP) system interconnecting heterogeneous medical devices [25], (2) a Petri Net research study of a network of nurse’s stations [40], and (3) a mock database research study for a point-of-care assistant [41].

The *Surgical Tool* term is a category that includes three surgical-device studies: (1) a scrub nurse robot [5], (2) a formal methods study of robot-surgery movement [33], and (3) a MDPnP tracheotomy control system [29].

Device names in Table 4 that do not match the query terms in Table 2 were taken from the actual research studies. For example, the combination *Resuscitator & Infusion Pump* is due to the WRAIR Computer Assisted Resuscitation Algorithm (CARA) medical device returned from the digital library search-term infusion pump.

RQ1: Our findings show that 4 of the 15 software-controlled medical devices found in FDA Class I recalls (listed in Table 1) have been investigated in existing software-testing research studies. The medical devices investigated are: (1) Infusion pumps, (2) Pacemakers, ICDs, or Defibrillators, (3) Surgical tools, and (4) Ventilators.

RQ1 Observation. The vast majority of the studies are related to the FDA Infusion Pump Initiative [12] or the Pacemaker Formal Methods Challenge [27]. Testing studies, initiatives, or challenges of other devices (anesthesia devices, central apnea ventilators, circulatory pumps, pharmaceutical compounders, cortical stimulators, continuous glucose meters, hemodialysis vasodilators, medical workstations, and general medical software) could be used to improve these devices.

For RQ1a, we investigate whether researchers tested an actual medical device or a simulated medical device. Using an actual device is often preferable; however, difficulty in obtaining expensive, proprietary, or yet-undeveloped devices requires the use of simulated device. Knowing the studies that used actual devices increases understanding of the limitations of the studies, applicability to other devices, and the potential areas for future research.

RQ1a: Our findings show that 15 of the 31 research studies have used actual medical devices.

RQ1a Observation. The research studies using actual medical devices are listed in Column (‘Research Studies’) of Table 4 in bold font. They are StateFlow[®] [24] and Hospira LifeCare[®] [28] infusion pumps, a Lego[®] Mindstorm surgical instrument tracking system [23], Burr Proton Therapy Center [34], Pacemaker Formal Methods Challenge [27] code [2, 16, 19, 20, 30, 31, 35], and CARA [3, 6, 10].

A third research study uses code “roughly based on the Abbott LifeCare[®] PCA3” [38] and we assume that it is not actual infusion pump code.

For RQ1b, we investigate whether research studies target a medical or a software engineering audience. Whether an article is published in a medical venue or a software engineering venue indicates the main audience for the research.

RQ1b: Our findings show that 3 out of 31 of the papers have been published in medical related journals: *Biomedical Instrumentation and Technology* [37], *Medical Imaging 2009; Visualization, Image-Guided Procedures, and Modeling* [23], and *International Conference of the IEEE Engineering in Medicine and Biology Society* [40].

RQ1b Observation. Research studies involving application of advanced software-testing techniques on medical devices are predominantly targeted towards a software engineering audience. This targeting may be due to the journal, conference, and workshop preferences; however, at least three medical venues accepted research studies on software-controlled medical devices. Thus, medical device engineers should consider software engineering venues, and/or software engineers should consider publishing in medical venues.

5. TYPES OF SOFTWARE-TESTING TECHNIQUES

The second group of questions that we investigate in this SLR

Table 5: Software-Testing Techniques

Infusion Pump	Formal Methods Modeling [21, 24], Model-Driven Development [36], Model-Based Development [36], Safety Case [42], Model Checking [38], Model-Based Dependability [38]
Medical Imaging	Regression Testing [23], Dependability Case [34], Product-Line Modeling [18], Safety Model [39]
Pacemaker / Design	Model Checking [2], Model-Driven Design [35], Assurance Case [19], Formal Methods Modeling [16, 22, 31], Model-Driven Development [30], Fault Injection [4], Timed Automata [8, 20]
Platform	Model-Based Development [25], Model Checking [40], Mock Object [41]
Resuscitation & Infusion Pump	Finite State Machine [3, 10], Automatic Scenario Generation [6]
Surgical Tool	Timed Automata [5], Formal Methods [33], Model Checking [29]
Ventilator	Digital Mock-up [32]

examines what types of advanced software-testing techniques were investigated in the studies. The techniques’ strengths and weaknesses are good indicators of the nature of failures discoverable by them. To minimize the bias by this SLR and the problem of describing an entire research study as a single technique, the testing technique term is searched, in order, from the research-study paper’s keywords, abstract, and then its introduction. We categorized the techniques into a somewhat arbitrary grouping to examine which techniques are applied to devices. We categorized automata as Timed Automata and Finite State Machine. Dependability is categorized as Safety Case, Assurance Case, and Dependability Case. We categorized all forms of modeling together: Formal Methods Modeling, Model-Based Design, Model-Driven Development, Model-Based Development, Model Checking, and Product-Line Modeling.

RQ2: Our categorization of the advanced software-testing techniques used in research studies (shown in Table 5) results in 18 Modeling, 5 Automata, 3 Dependability, 2 Mocking, and 1 each of Automatic Scenario Generation, Regression Test, and Fault Injection.

RQ2 Observation. Most research studies describe their technique as some form of modeling. Given that testing cannot assert the absence of faults and that each technique has advantages and disadvantages, providing more varied advanced testing techniques would improve software-controlled medical devices.

For RQ2a, we examine whether a technique is specific to a device type. In particular, we investigated whether testing techniques differed between device types.

RQ2a: Table 6 shows that infusion pumps and the pacemaker ICDs were the focus of the majority of modeling research studies. The other device types had no more than two research studies per category.

RQ2a Observation. Modeling is the predominant advanced software-testing technique for two of the device types. Since our somewhat arbitrary categorizations grouped all formal methods together in the modeling category, and since the Pacemaker Formal Methods Challenge [27] encouraged some research studies, we were not sur-

Table 6: Software-Testing Techniques by Category

Infusion Pump	6 Modeling [21, 24, 28, 36, 37, 38] 1 Dependability [42]
Medical Imaging	2 Modeling [18, 39] 1 Regression Testing [23] 1 Dependability [34]
Pacemaker / ICD	6 Modeling [2, 16, 22, 30, 30, 35] 2 Automata [8, 20], 1 Dependability [19] 1 Fault Injection [4]
Platform	2 Modeling [25, 40], 1 Mocking [41]
Resuscitation & Infusion Pump	2 Automata [3, 10] 1 Automatic Scenario Generation [6]
Surgical Tool	1 Automata [5], 2 Modeling [29, 33]
Ventilator	1 Mocking [32]

prised by the number of modeling studies of pacemakers. However, it is not clear whether the FDA Infusion Pump Initiative [12] encouraged the modeling research studies on infusion pumps. Maybe the modeling techniques applied to infusion pumps and pacemakers apply equally to all other medical devices. Or it may be device types like ventilators, surgical tools, and others require their own modeling research studies. The lack of other techniques on other device types may indicate areas of future research.

6. IMPROVEMENT OF MEDICAL DEVICES

The third question that we investigate is how advanced software-testing techniques improved software-controlled medical devices. Are the causes of the failures found in FDA Class I recalls known? Could the testing techniques detect faults that cause the types of these failures? Could the testing techniques be used without having access to the source code of the software under test?

RQ3: *In order for advanced software-testing techniques to improve software-controlled medical devices, the following questions have to be true: RQ3.1: Did recalls provide sufficient information to help identify causes of the failures? RQ3.2: Could advanced software-testing techniques detect faults that cause these types of failures? RQ3.3: Which advanced software-testing techniques could be potentially used by regulators (i.e., without requiring access to source code of the software under test)?*

For RQ3.1, we investigate whether regulators specified sufficient information to identify causes of failure for use in research studies. Of the software related FDA Class I recalls, some specified only information to identify the failure but nothing further. For instance we found that a recall that specifies only ‘this device may fail to sound an alarm’ ([ucm152725.htm](http://www.fda.gov/ucm152725.htm))³ describes the failure to users and caregivers; however, the recall description fails to provide information as to why the device fails to sound an alarm or any identification of the cause of the failure. For some FDA Class I recalls, the failure cause cannot be identified but some secondary information can be used to present a hypothesis. For example, a recall description specifies ‘the computer may shut down (stop pumping) without an alarm’ ([ucm203872.htm](http://www.fda.gov/ucm203872.htm)). It is reasonable for researchers to assume a requirement that all abnormal shutdowns require an alarm and a failure is observed whenever the requirement is not met.

³FDA recalls can be found at <http://www.fda.gov/MedicalDevices/Safety/ListofRecalls/> postfixed with the `ucmxxxxx.htm` identifier.

Table 7: Technique Categorization

Infusion Pump	Gray-box [21, 28, 36, 38] Black-box [24, 37, 42]
Medical Imaging	White-box [23] Gray-box [18, 34] Black-box [39]
Pacemaker / ICD	White-box [2, 8, 30, 31] Gray-box [4, 16, 19, 20] Black-box [22, 35]
Platform	Gray-box [25, 41] Black-box [40]
Resuscitation & Infusion Pump	Gray-box [6, 10] Black-box [3]
Surgical Tool	Gray-box [29] Black-box [5, 33]
Ventilator	Black-box [32]

RQ3.1: *Of the software related FDA Class I recalls, 77% had sufficient information to help identify causes of failure for use in research studies.*

RQ3.1 Observation. Eliminating the *Non-software* recalls from Table 1, we found that most recalls provide some information about the cause of the medical device failure. In general, providing more information about the failure cause makes it more likely that an advanced software-testing technique could expose the failure.

For RQ3.2, we investigate whether any advanced software-testing technique could detect faults that cause the types of the failures found in FDA Class I recalls. For the software related FDA Class I recalls of Table 1, each research study is numerically ranked against every recall with a potential cause: Yes (2), Maybe (1), or No/Unknown (0). For some recalls, the cause of the failure may be obvious even though none of the techniques in the 31 studies could detect the corresponding fault. For example, ‘the entry of hours into the minute field’ ([ucm064764.htm](http://www.fda.gov/ucm064764.htm)) is a user error that could not be detected by any of the techniques in the selected studies.

RQ3.2: *Of the software related FDA Class I recalls that specify a potential failure cause from RQ3.1, faults corresponding to 50% of the failure types had some chance of getting identified by an advanced software testing technique.*

RQ3.2 Observation. Examining the software related recalls that have a potential failure cause, we found that most of the recall failure causes may be identified by one or more of the techniques. If proprietary information could be restricted to developers and regulators while allowing researchers access to the failure cause information, the chances of a technique being developed to identify the cause of the failure would increase.

For RQ3.3, we examine whether the source code is required to use the techniques. Unless a manufacturer makes the code generally available through a challenge, only developers have source code access.

White-box testing relies on the internal structure of the software while black-box testing is independent of it. White-box testing requires source code access. For this SLR, we consider gray-box testing to be from the end-user perspective. Gray-box testing requires some internal knowledge during test design but is otherwise similar to black-box testing. Table 7 shows the categorization of testing techniques with white-box testing conducted by only developers while black and gray-box testing can be conducted by most developers, regulators, and researchers.

RQ3.3: 27 out of 31 research studies used testing techniques that requires no access to source code of the software under test. The white-box testing techniques that require source code access are shown in Table 7 and include Albarghouthi [2], David [8], Jomier [23], Méry1 [30], and Méry2 [31].

RQ3.3 Observation. With some internal knowledge of the medical device design, most advanced software-testing techniques can generally be conducted by developers, regulators, and researchers. Developers and regulators can provide some internal information of the devices to researchers to further advance testing-technique research.

7. TESTING TECHNIQUES AND REPORTED FAILURES

For the forth question, we compare the research-study technique categories of Table 5 and Table 7 against the FDA recall categories in Table 1 to see how well the research techniques overlap with the commonly occurring types of failures. Note that the *platform* medical device studies are not included for this question. Two *platform* studies are for a conceptual medical platform [25, 40] and one *platform* is for a research study on testing against a mock database [41]. Furthermore, the *CARA Resuscitation & Infusion Pump* [3, 6, 10] medical device is not publically available and has no recalls, and hence it is not included in this question.

RQ4 Infusion pump: Our findings indicate that infusion pump research studies cover the larger groups of failure types of FDA Class I infusion pump recalls: software alarms, user-interface issues, and medical dose calibration.

RQ4 Infusion Pump Observation. From Table 1, among 31 infusion pump recalls, *general software* and *alarm failures* are the largest category. Two of the five infusion pump studies investigate *software alarms* [21, 28]. User interface issues are the next most common and these failures are exposed by *user-centered design* [28] and *all possible buttons and screens* [21] and a *human interaction model* [38]. *Conversion errors* [38] and *GUI model for operator errors* [37] are also covered. More than a quarter of the infusion pump failure types fall in the medication dose calibration category. One research study investigates how to test for under-infusion errors [37] while another research study proposes techniques to verify that infusions stop if the monitor observes a failure [28].

RQ4 Medical Imaging: Our findings are inconclusive in determining whether medical imaging research studies cover the commonly occurring failure types of FDA Class I medical imaging recalls.

RQ4 Medical Imaging Observation. FDA separates out medical device recalls [13] from radiation-emitting products recalls [14]. Unfortunately, the Class I recalls from the radiation-emitting site cover only 500 recalls back to 2003, which does not overlap with the same time frame as the medical device research studies. In particular, one of the medical imaging studies occurred in 2001 [39] prior to the recall time frame. Additionally, the FDA site incompatibilities include differences in recall classification. For example, a radiation-emitting product recall for a respirator mask appeared as three separate recalls; one each for small, medium, and large [11].

RQ4 Pacemaker / ICD: Our findings indicate that FDA recalls for pacemaker and ICD software alarms were covered by the advanced software-testing techniques in the pacemakers and ICDs studies.

RQ4 Pacemaker / ICD Observations. Among the 22 Class I defibrillator recalls in Table 1, the largest category of recall is *general software failures*, with nearly as many *mechanical or electrical failures*. The third largest category is *software alarm failures* that range from messages not being visible, message timing issues, and generic “DEFIB COMM” messages. The modeling pacemaker ICD research studies [2, 16, 22, 30, 31, 35] and the automata research studies [8, 20] cover these respective areas.

RQ4 Surgical Tool: Our findings could show only one correspondence between failure types of FDA recalls and failure types of surgical-tool research studies.

RQ4 Surgical Tool Observation. One recall description of a surgical handpiece refers to a switch that ‘self-activates without pushing the trigger, continues running after releasing the trigger, and runs in unintended directions’ (*ucm181784.htm*), which is very similar to the failure type focus of the research study for a tracheotomy tool that locks out a ventilator supplying oxygen when the laser is triggered [29].

RQ4 Ventilator: Our findings could not show any correspondence between FDA recalls and the ventilator study.

The sole ventilator study is a mock device study [32]. It is unclear how we could relate the study to the 14 ventilator recalls in Table 1.

8. CONCLUSION

The goal of our work presented in this paper was to synthesize the available research studies and FDA Class I recalls to gain insight into the current state-of-the-art software-controlled medical devices testing.

Commonly found failure types described in recalls overlap with failure types covered by testing techniques. For the infusion pump, a small number of techniques (modeling and dependability) focus on the failure types most commonly found in FDA recalls (alarms, user-interface, and medical dose calibration). For the pacemaker ICD, more techniques (modeling, dependability, automata, fault injection) focus on a smaller category of commonly found failure types (alarms and general timing). A surgical tool study overlaps a single recall. In our opinion, while investigating future research-studies choices, the number of existing studies of a device types is not a good indicator of whether additional research studies are needed.

The diversity of techniques and the study audience may be an indication for the need of additional research studies. Most of the studies are related to some form of modeling techniques. A more varied set of advanced software-testing techniques applied to medical devices may be desirable. Targeting studies for medical as well as software engineering venues increases the audience reading the papers, likely further improving software-controlled medical devices. Future research study considerations should include the range of existing study techniques and audiences.

Additional recall failure information would improve research stud-

ies. For software related recalls that specify a potential cause, the existing techniques in the studies have some chance of identifying the likely cause of the failure. By limiting the analysis to only those recalls that describe potential causes, or by examining techniques that have *some* chance of exposing the failure, there is room for improvement. More standardized causal information in recalls enables higher quality artifacts to researchers, which in turn would produce more research studies. Having a *National Transportation Safety Board type database of medical device software failures would provide researchers with the most information* [17] to improve software-controlled medical devices.

Additional recall information would help developers as well as researchers. Providing more recall information not only helps researchers, but also helps developers. The analysis of whether any technique would expose the failure is not sufficient for developers with limited resources. It is unreasonable to assume all or most techniques would be applied during device development. Having more causal information facilitates developer understanding of previous failures and how those failures may relate to the device being developed. The additional information on the failure cause also facilitates choices of which techniques to apply during development to increase the chances of early identification of commonly occurring failure types.

Initiatives and challenges facilitate more research. Even without additional proprietary information, researchers are providing insight into medical-device software safety. The FDA Infusion Pump Initiative [12] and the Pacemaker Formal Methods Challenge [27] provide researchers with artifacts to examine and directions to explore. With 7 infusion pump studies and 10 pacemaker studies, these two device types have the most number of research studies and the most direct correlation to FDA recalls.

As software controls in medical devices evolve from monolithic devices to cyber-physical systems the direction of future research studies is important. Testing techniques must improve to expose failures in more complex MCPS. The use of initiatives and grand challenges may be helpful in shaping a future research direction. Additionally, non-proprietary failure recall information would provide useful artifacts used in research studies. In light of aforementioned artifacts and with more advanced software-testing technique being studied, software-controlled medical device are expected to further improve.

ACKNOWLEDGMENT

We would like to thank Dr. Yi Zhang of Center for Device and Radiological Health at FDA for helpful discussions and suggestions. This work is supported in part by NSF grants CCF-0845272, CCF-0915400, CNS-0958235, CNS-1160603, an NSA Science of Security Lablet grant, a NIST grant, a Microsoft Research Software Engineering Innovation Foundation Award.

References

- [1] 21 CFR 803.20. Code of Federal Regulations Title 21 Part 803.20 @ONLINE. <http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?fr=803.2>, Apr. 2012.
- [2] A. Albarghouthi, A. Gurfinkel, and M. Chechik. Whale: an interpolation-based algorithm for inter-procedural verification. In *Proc. International conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pages 39–55, 2012.
- [3] R. Alur, D. Arney, E. Gunter, I. Lee, J. Lee, W. Nam, F. Pearce, S. Van Albert, and J. Zhou. Formal specifications and analysis of the computer-assisted resuscitation algorithm (CARA) infusion pump control system. *International Journal on Software Tools for Technology Transfer (STTT)*, 5(4):308–319, 2004.
- [4] H. Ammar, S. Yacoub, and A. Ibrahim. A fault model for fault injection analysis of dynamic UML specifications. In *Proc. International Symposium on Software Reliability Engineering (ISSRE)*, pages 74–83, 2001.
- [5] A. Anier and J. Vain. Timed Automata based provably correct robot control. In *Proc. Biennial Baltic Electronics Conference (BEC)*, pages 201–204, 2010.
- [6] M. Auguston, J. Michael, and M. Shing. Environment behavior models for automation of testing and assessment of system safety. *Information and Software Technology*, 48(10):971–980, 2006.
- [7] I. S. C. Committee et al. IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). CA: IEEE Computer Society, 1990.
- [8] A. David, M. Möller, and W. Yi. Formal verification of UML statecharts with real-time extensions. In *Proc. Fundamental Approaches to Software Engineering*, pages 208–241. Springer, 2002.
- [9] R. Dolores 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*, 8(04):351–371, 2001.
- [10] D. Drusinsky, M. Shing, and K. Demir. Creation and validation of embedded assertion statecharts. In *Proc. International Workshop on Rapid System Prototyping*, pages 17–23, 2006.
- [11] FDA. Class 1 Recall ComfortGel Nask Mask @ONLINE. <http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm?id=29870,29871,29872>, Dec. 2003.
- [12] FDA. White Paper: Infusion Pump Improvement Initiative @ONLINE. <http://www.fda.gov/medicaldevices/productsandmedicalprocedures/GeneralHospitalDevicesandSupplies/InfusionPumps/ucm205424.htm>, Apr. 2010.
- [13] FDA. List of Device Recalls @ONLINE. <http://www.fda.gov/MedicalDevices/Safety/RecallsCorrectionsRemovals/ListofRecalls/default.htm>, Jan. 2013.
- [14] FDA. Medical & radiation emitting device recalls @ONLINE. <http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm>, Jan. 2013.
- [15] B. Glaser and A. Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Aldine de Gruyter, 1967.
- [16] A. Gomes and M. Oliveira. Formal development of a cardiac pacemaker: from specification to code. *Formal Methods: Foundations and Applications*, pages 210–225, 2011.
- [17] D. Jackson. A direct path to dependable software. *Comm. of the ACM*, 52(4):78–88, 2009.
- [18] M. Jaring, R. Krikhaar, and J. Bosch. Modeling variability and testability interaction in software product line engineering. In *Proc. 7th International Conference on Composition-Based Software Systems (ICCBSS)*, pages 120–129, 2008.

- [19] E. Jee, I. Lee, and O. Sokolsky. Assurance cases in model-driven development of the pacemaker software. *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 343–356, 2010.
- [20] E. Jee, S. Wang, J. Kim, J. Lee, O. Sokolsky, and I. Lee. A safety-assured development approach for real-time software. In *Proc. IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 133–142. Springer, 2010.
- [21] R. Jetley, S. Purushothaman Iyer, and P. Jones. A formal methods approach to medical device review. *Computer*, 39(4):61–67, 2006.
- [22] Z. Jiang, M. Pajic, A. Connolly, S. Dixit, and R. Mangharam. Real-time heart model for implantable cardiac device validation and verification. In *Proc. 22nd Euromicro Conference on Real-Time Systems (ECRTS)*, pages 239–248, 2010.
- [23] J. Jomier, L. Ibanez, A. Enquobahrie, D. Pace, and K. Cleary. An open-source framework for testing tracking devices using Lego Mindstorms. In *Proc. International Society for Optics and Photonics (SPIE) Medical Imaging*, pages 1–7, 2009.
- [24] B. Kim, A. Ayoub, O. Sokolsky, I. Lee, P. Jones, Y. Zhang, and R. Jetley. Safety-assured development of the GPCA infusion pump software. In *Proc. International Conference on Embedded Software (EMSOFT)*, pages 155–164, 2011.
- [25] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, and S. Weininger. An open test bed for medical device integration and coordination. In *Proc. International Conference on Software Engineering (ICSE) Companion Volume*, pages 141–151, 2009.
- [26] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33:2004, 2004.
- [27] S. Q. R. Laboratory. Pacemaker Formal Methods Challenge @ONLINE. <http://sqr1.mcmaster.ca/pacemaker.htm>, Apr. 2007.
- [28] I. Lee, O. Sokolsky, S. Chen, J. Hatcliff, E. Jee, B. Kim, A. King, M. Mullen-Fortino, S. Park, A. Roederer, et al. Challenges and research directions in medical cyber-physical systems. *IEEE*, 100(1):75–90, 2012.
- [29] T. Li, F. Tan, Q. Wang, L. Bu, J. Cao, and X. Liu. From offline toward real-time: A hybrid systems model checking and CPS co-design approach for Medical Device Plug-and-Play (MDPnP). In *Proc. International Conference on Cyber-Physical Systems (ICCPs)*, pages 13–22, 2012.
- [30] D. Méry and N. Singh. A generic framework: from modeling to code. *Innovations in Systems and Software Engineering*, 7(4):227–235, 2011.
- [31] D. Méry and N. Singh. Formalization of heart models based on the conduction of electrical impulses and cellular automata. *Foundations of Health Informatics Engineering and Systems*, pages 140–159, 2012.
- [32] B. Miller, F. Vahid, and T. Givargis. Application-specific codesign platform generation for digital mockups in cyber-physical systems. In *Proc. Electronic System Level Synthesis Conference (ESLsyn)*, pages 1–6, 2011.
- [33] R. Muradore, D. Bresolin, L. Geretti, P. Fiorini, and T. Villa. Robotic surgery. *IEEE Robotics & Automation Magazine*, 18(3):24–32, 2011.
- [34] J. Near, A. Milicevic, E. Kang, and D. Jackson. A lightweight code analysis and its role in evaluation of a dependability case. In *Proc. International Conference on Software Engineering (ICSE)*, pages 31–40, 2011.
- [35] M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, and R. Mangharam. From verification to implementation: A model translation tool and a pacemaker case study. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 173–184, 2012.
- [36] A. Ray, R. Jetley, and P. Jones. Engineering high confidence medical device software. *ACM SIGBED Review*, 6(2):1–7, 2009.
- [37] A. Ray, R. Jetley, P. Jones, and Y. Zhang. Model-Based Engineering for Medical-Device Software. *Biomedical Instrumentation & Technology*, 44(6):507–518, 2010.
- [38] S. Sankaranarayanan, H. Homaei, and C. Lewis. Model-based dependability analysis of programmable drug infusion pumps. In *Proc. Formal Modeling and Analysis of Timed Systems*, pages 317–334. Springer, 2011.
- [39] K. Sayre, J. Kenner, and P. Jones. Safety models: An analytical tool for risk analysis of medical device systems. In *Proc. Computer-Based Medical Systems (CBMS)*, pages 445–451, 2001.
- [40] E. Sloane and V. Gelhot. Applications of the Petri net to simulate, test, and validate the performance and safety of complex, heterogeneous, multi-modality patient monitoring alarm systems. In *Proc. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, volume 2, pages 3492–3495, 2004.
- [41] K. Taneja, Y. Zhang, and T. Xie. MODA: Automated test generation for database applications via mock objects. In *Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 289–292, 2010.
- [42] S. Zafar and R. Dromey. Integrating safety and security requirements into design of an embedded system. In *Proc. Asia-Pacific Software Engineering Conference (APSEC)*, pages 8–17, 2005.