Requirement Engineering for Functional Alarm System for Interoperable Medical Devices

Krishna K. Venkatasubramanian[1], Eugene Y. Vasserman[2], Vasiliki Sfyrla[3], Oleg Sokolsky[4], and Insup Lee[4]

[1] Worcester Polytechnic Institute, kven@wpi.edu
[2] Kansas State University, eyv@ksu.edu
[3] Unaffiliated, vasssiliki@gmail.com
[4] University of Pennsylvania, {sokolsky,lee}@cis.upenn.edu

Abstract. This paper addresses the problem of high-assurance operation for medical cyber-physical systems built from interoperable medical devices. Such systems are different from most cyber-physical systems due to their "plug-and-play" nature: they are assembled as needed at a patient's bedside according to a specification that captures the clinical scenario and required device types. We need to ensure that such a system is assembled correctly and operates according to its specification. In this regard, we aim to develop an alarm system that would signal interoperability failures. We study how plug-and-play interoperable medical devices and systems can fail by means of hazard analysis that identify hazardous situations that are unique to interoperable systems. The requirements for the alarm system are formulated as the need to detect these hazardous situations. We instantiate the alarm requirement generation process through a case-study involving an interoperable medical device setup for airway-laser surgery.

1 Introduction

A recently emerging vision of dynamically composable and Interoperable Medical Device Systems (IMDS) will allow information integration from multiple clinical sources in a context-sensitive way to guide patient care and prevent common critical mistakes [12]. Various agencies and standards bodies, including the U.S. Food and Drug Administration, have signaled that the future of medical technology lies in medical device interoperability [12]. High-assurance device interoperability will be a critical requirement in realizing this vision.

IMDS, while a subset of cyber-physical systems in general, are unique in that they are constructed as needed, i.e., *on demand*. An interoperability failure can lead to devastating consequences for the patients being treated e.g., sudden loss of closed-loop control. These systems are mission-critical — literally a matter of life and death — therefore we must ensure they are functioning (availability) and functioning properly (correctness) with high assurance. Risks in medical device interoperability arise due to its dynamic nature, meaning *the potential issues are emergent properties of the entire system* and therefore impossible to

fully control ahead of time, so we need to monitor the "health" and proper functionality of the system itself at run-time. This means continually verifying that design-time assumptions hold at run-time and ensuring that faults, either natural or malicious, are detected.

IMDS typically consists of two main entitiy classes: medical devices involved in the treatment of patients, and software controller applications (referred to as *apps*) that coordinate these devices. An illustrative example is airway-laser surgery, in which a surgeon uses a laser to investigate or fix issues with the patient's trachea or nearby organs. This carries the potential danger of a fire if the laser is active while oxygen is being delivered to the patient and the surgeon accidentally damages the oxygen delivery tube. Whenever the laser is being activated, a human operator must first block the air path. In a simplified patient model, brain damage may occur if oxygen is reduced for more than about 4.5 minutes or the blood oxygen saturation (SpO2) level drops below 40% — both of these conditions must be avoided. In traditional operating room environments, nurses and surgeons are supposed to be aware of such potential fire and low-oxygen problems [10].

When an IMDS for Airway-Laser-Surgery (ALS-IMD) is deployed, it needs to ensure that interactions between constituent devices proceed according to the specification of the clinical scenario. Otherwise, the safety requirements of the system can be compromised, even if the controller itself is operating properly. If problematic interactions are detected, an alarm needs to be raised. Alarms are therefore a key component in ensuring safe operation of IMDS. Note that, these alarms are different from the clinical alarms in that they do not signal abnormal patient state, but rather the abnormal state of the entities that monitor and treat the patient. We refer to these as *technical alarms* (rather than *physiological or clinical alarms* which have to do with patient health).

In this paper, we explore the requirements to realize a robust yet flexible alarm system for IMDS. Henceforth, we refer to such a system as the *Interoperability Alarm System (IAS)*. The goal is to reliably alert clinicians to failures of the overall interoperability of the system. We use the term *fault* to mean cause(s) of *error* in the system, which may then eventually lead to loss of expected service from the system, or *failure* [3]. The main challenge in developing these requirements is that the IAS needs to support all IMDS that can be assembled from the available set of interoperable devices. (The full magnitude and variability of this set may not be known at IMDS design time.) Our approach to developing the requirements for IAS has three steps: (1) perform a hazard analysis for identifying interoperability failures within ICE, (2) identify the various faults that individual elements in the IMDS can manifest, (3) derive a set of fault trees that characterize how combining various faults may lead to identified hazards. Note that we do not claim that the use of hazard analysis or fault-trees to characterize the failures of IMDS is by itself novel. Novelty comes from applying existing and well-understood tools to the new and unique domain of IMDS.

In this regard, we consider an IAS that is decoupled from other entities in IMDS: although the controller app and medical devices are well-defined in

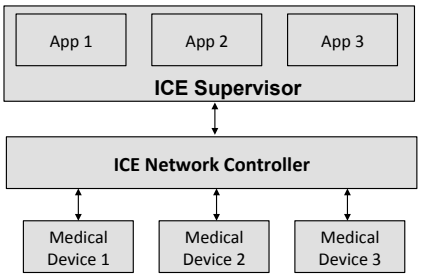

Fig. 1. Simplified diagram of the ICE architecture

terms of intended use and expected behavior, they cannot reliably monitor their own operation and trigger alarms. A key feature of the IAS is the flexibility of connection-time (i.e., system start-time) configuration, using specifications (required devices, information flows and their parameters) supplied by deployed apps and parameters of interoperability-compliant devices used to build the IMDS. Stand-alone medical devices already have built-in alarms, but in the interoperable context, augmenting device alarms with a high-level system alarm will not only simplify medical device design but will also increase their safety.

The **contributions** of this work are as follows: (1) a hazard analysis based requirements generation for IAS, and (2) a case study using the airway-laser surgery example to some of the demonstrate the utility and expressiveness of our approach.

2 Background

Within the realm of healthcare interoperability, much work is done at the syntactic level (i.e., developing common formats for data exchange between medical systems [14]) in the form of standards such as 11073 [4], IHE [9] and HL7 [6]. However, we believe that realizing the full potential of IMDS needs a higher level of interoperability. Hence we make use of the ASTM 2761 Integrated Clinical Environment [2] standard, also known as MD PnP ICE, which aims to provide semantic interoperability (i.e., devices not only have a common data format, but they also understand the meaning of the data being exchanged). We adopt the ICE standard as the primary *system model* for our work. Logically, the ICE architecture is separated into three entities, which are illustrated in Figure 1:

Supervisor (SUP): The Supervisor is responsible for executing clinical "scenarios" also referred to as *apps*, from common and easily scriptable tasks such as taking blood pressure at predefined intervals and recording the results, to more complex procedures like medication interaction monitoring and suppression of likely false alarms. The app thus encapsulates clinical knowledge about a particular treatment procedure that involves multiple devices. These clinical scenarios are viewed as control algorithms running on the Supervisor.

Medical Device (MDs): Medical devices in the ICE setup can be responsible for measuring the state of the patient or changing the state of the patient

through some form of stimulus (e.g., electrical, mechanical) with the express object of causing a particular outcome (e.g., pace the heart, reduce blood glucose).

Network Controller (NC): ICE allows coordination between each medical device through the Network Controller, a sort of "medical router". It does not have any medical/clinical functionality itself, but facilitates communication between the medical devices and the Supervisor, with responsibilities including data routing, translation, and quality of service (QoS) enforcement.

In this architecture, the Supervisor and the NC are fixed, while MDs and apps are dynamic and change based on the clinical scenario. In [11], the authors describe a means for each medical device to provide a model of its operation during run-time to the app. The apps too provide a specification of their operation to the Supervisor to ensure correct operation. We assume these models and specifications are also made available at system startup to the IAS, which can then use them to determine the presence of ICE failures. In addition the IAS is already assumed to possess a model for the Supervisor the NC, which is pre-supplied and does not change based on clinical scenarios. The availability of these models and specifications make it possible for the IAS to have a global view of the IMDS operation and therefore potentially detect failures that IMDS cannot detect by itself.

2.1 A case for IMDS alarms

Currently, medical systems are designed and deployed as monolithic, complete artifacts [5]. Only a small number of vertically-integrated systems permit any deviation from the original configuration. Integrators build systems for a specific clinical scenario out of a dedicated collection of devices, and then argue for the safety of this system. In this case, we can craft a monitoring subsystem that would raise alarms based on the requirements of the clinical scenario and failure modes of the devices involved. A fixed safety argument based on hazards of the clinical scenario can be built. A dynamically composed setup, on the other hand, is customized (usually by the appointed clinician) for each patient for whom it is deployed in terms of the included devices and applications that run on it, and **the monolithic system safety approach does not straightforwardly extend to the on-demand interoperability setting**, since the safety argument is no longer tied to a particular scenario, nor are the "safe" parameters known ahead of time. Crafting a safety argument ahead of deployment time for a system is challenging, to say the least. However, a high-assurance alarm subsystem can significantly simplify the task, if we can ensure that unsafe conditions reliably trigger an alarm and a failover-to-operator (manual, non-interoperating) mode.

Stand-alone medical devices already incorporate their own alarm systems. Indeed, alarm parameters, e.g., the concept and quantification of deviation from "safe," vary from patient to patient, and may even differ for the same patient for different treatment strategies. For instance, if a patient receives blood pressure reducing medication, "safe" / normal readings are expected to be outside a predefined range — this information is known to the clinician, but potentially not to the alarm system. Clinical logic (automatic reasoning about the condition of the

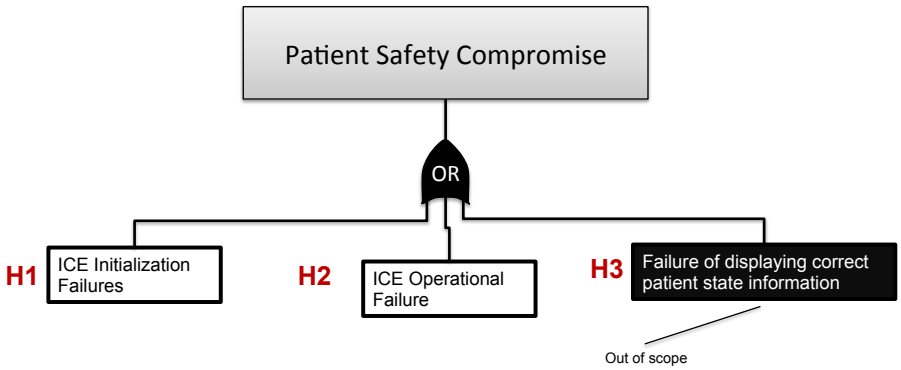

Fig. 2. Hazard causing patient safety

patient) can be almost arbitrarily complex, with a large number of factors, from basic patient information like height, weight, and gender, to current medications and condition treated. ICE apps are expected to receive data about the same patient from different sources, and perform a clinically useful function. One such function would be monitoring patient vital signs and predicting potential problems, which would be manifested as alarms. We call these patient health alarms, or *physiological or clinical alarms*. Since apps are the components responsible for raising clinical alarms, we consider such alarms out of scope for this paper. However, apps rely on correct operation of medical devices and the underlying ICE platform. Which faults may lead to failures depends on which apps are deployed in the ICE instance. No single component within the system has all the relevant information to make informed decisions regarding which faults to detect and whether or not to raise an alarm in the event of a fault. Therefore, we focus on **technical alarms** that are the *result of a fault of the ICE instance* — emergent issues that arise because of the very fact that the system is interoperable and constructed as needed from units whose functionality, clinical logic, and domain data are limited.

3 IAS Requirements Generation

In this section we describe requirements generation for the Interoperability Alarm System (IAS) — the technical alarm system for IMDS. The IAS is responsible for detecting any interoperability failures within IMDS itself; in our case this is ICE. A well-designed ICE infrastructure should itself be capable of detecting technical issues with its operation, similar to exception handling in software engineering: if a medical device does not start when it is supposed to, or if the Supervisor crashes, such technical faults can be detected within ICE by e.g., the Supervisor or the device itself, respectively. Therefore the technical issues of ICE under the purview of IAS are those that cannot be reliably detected within ICE. Our main challenge is to determine when self-checks within ICE are insufficient.

Let us further consider our ongoing example of airway-laser surgery. A device, e.g., the pulse oximeter, sending valid but incorrect values is out of scope

for the moment, since this does not constitute an interoperability problem. On other hand, sending values at a wrong rate is an interoperability problem that, presumably, can be detected by the ICE component receiving the values. However, because of the on-demand nature of ICE, the receiver may not know what the right rate is (since we may use different oximeters in different instantiations). Thus, whoever checks the value has to match the expected rate against the rate provided by the specific device used in a particular instance. If the check is performed incorrectly, we are in a situation that requires detection by IAS.

Our general approach to identify the (categories of) requirements for IAS has three steps: (1) identifying the principal hazards in ICE, (2) determining the high-level causes for these hazards, and (3) identifying the basis of the causes in the previous step. We build a fault-tree for each of these ICE failures to trace the origins of the fault. The output of the fault-tree analysis gives us a set of conditions within ICE operation that can eventually lead to hazards and **failure of ICE that IAS is responsible for detecting**, hence forming its requirements. In other words, the leaf nodes of this fault-tree essentially provide us with a category of IAS requirements.

We now present a categorization of the IAS requirements rather than list each of them one by one, i.e., we do not attempt to provide a comprehensive list of all possible sources of hazards within ICE. Further, IMDS hazards and the faults leading to them are also situation-specific. The fault trees are therefore pruned appropriately.

3.1 Hazard Analysis-based Requirements Gathering

The goal of hazard analysis is to determine, based on the operational understanding of a system, the various situations which can lead to hazardous condition for its users. In the case of ICE, hazardous conditions pertain to situations where the interoperability capability provided by ICE is not executed or executed incorrectly, leading to patient safety consequences. For IMDS we see that patient safety compromise appears from a set of three main hazards in ICE (see Figure 2): (1) *Hazard H1:* ICE initialization failure, which prevents the interoperability platform from executing, leading to the patient not being able to receive its benefits; (2) *Hazard H2:* ICE operational failures, which occur during the executing of the interoperability by ICE, leading to sudden stoppage or incorrect interoperability; (3) *Hazard H3:* ICE status presentation failure, which leads to the wrong patient and ICE status being conveyed (e.g., through patient displays connected to ICE) to the caregiver and the patient, potentially leading to incorrect diagnosis and treatment. We consider H3 to be out of scope because if an app is displaying the wrong information (but acting on correct information, as the hazard specifies) there is a problem only if a clinician observes the information and acts on it. If the display did not exist, everything would be operating normally. We now describe the causes of in-scope hazards H1 and H2 in detail:
Initialization Failures: H1 focuses on problems within ICE during the initialization of the interoperability setup around the patient. These manifest themselves when the various medical devices are integrated with ICE and the apps

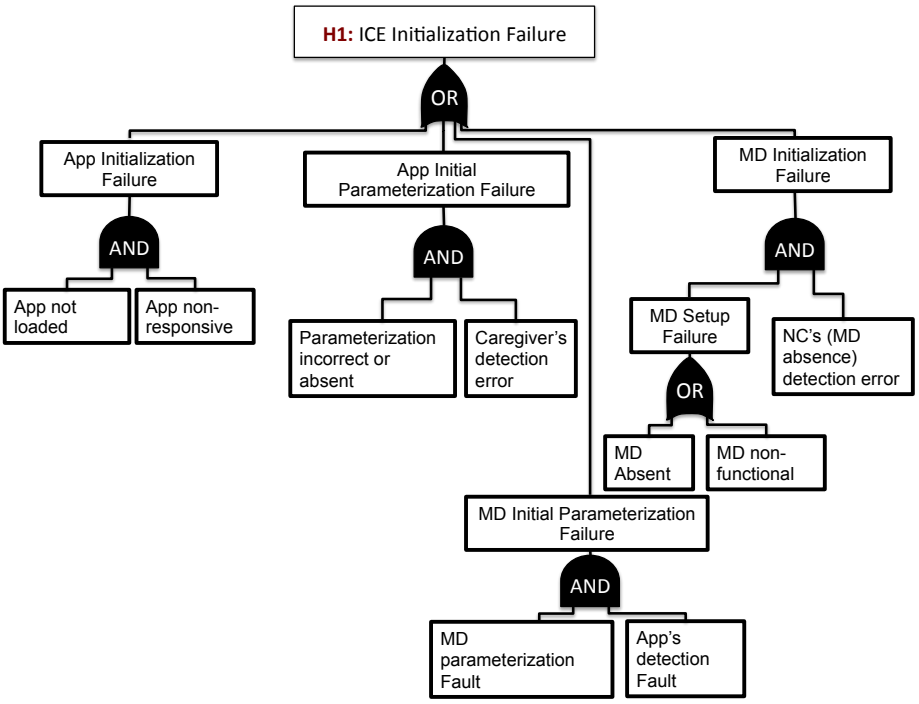

Fig. 3. Fault-tree for H1: Initialization failures

loaded onto the Supervisor within ICE and the interoperation is started. At this stage, failures can manifest themselves due to faults at all levels of ICE from the medical device to the Supervisor. For example, supervisory coordinator (i.e., app) initialization fault can occur if the Supervisor is not able to load the app. Similarly, the supervisor control algorithm can itself have initialization issues where it is not parameterized correctly or at all. Figure 3 illustrates the fault-tree for the initialization failures.

Operational Failures: H2 considers operational failures within the ICE, which happen after a successful (failure-less) initialization. The causes of H2 can again manifest themselves anywhere within ICE. An example of an operational failure is when a medical device sends corrupted data and the Network Controller does not check the CRC of the data received, failing to detect corruption. If either of the entities worked correctly, the fault would be detected within ICE and IAS would not be engaged. Other examples include: (1) a mismatch between actual medical device operation and operation expected by the app; (2) loss of device data or commands from the app due to insufficient bandwidth in the Network Controller; (3) software fault in the app crashing the Supervisor. Figure 4 illustrates the fault-tree for the operational failures. Some of the intermediate events appear more than once in the fault tree. For such events, we have expanded only one of the many occurrences into its constituent basic events. For the rest, we simply use a dashed line to denote that they should be expanded further.

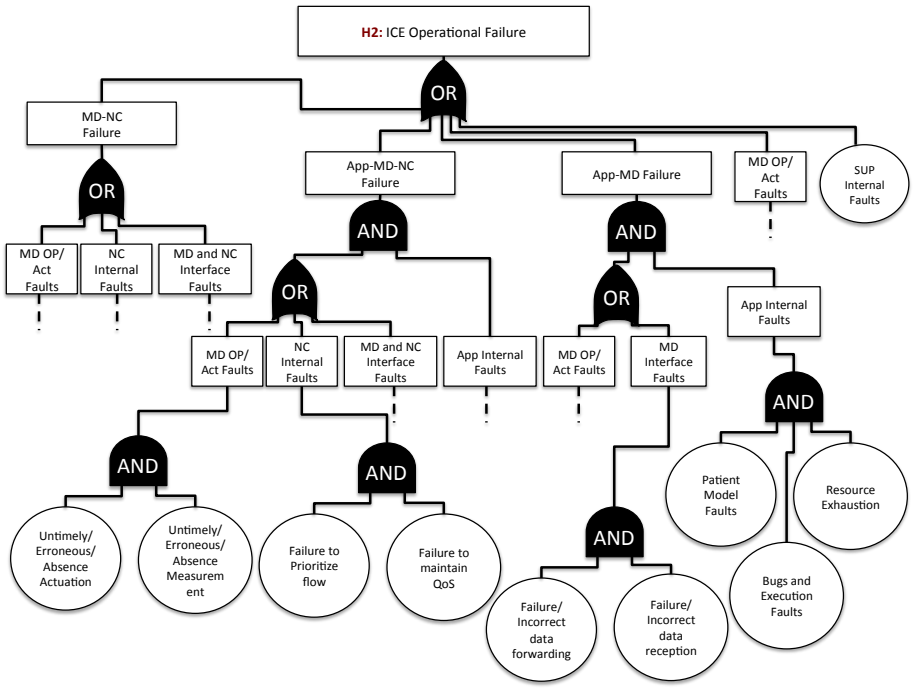

Fig. 4. Fault-tree for H2: Operational failures

As we can see from the fault-trees generated for the initialization and operational failures, failures of ICE that fall under the purview of IAS are those where one or more of the four components in the ICE fails to follow the prescribed protocol while at the same time the ICE entities fail to detect that the protocol is not being followed. For example, a medical device fault in measuring the patient's physiology is not by itself sufficient to involve the IAS. That is, IAS will be involved only if a fault in the data source ICE entity is accompanied by a fault in the data sink ICE entity (the entity ultimately receiving and processing the data). In general, there appear to be four types of failure scenarios in ICE operation that cannot be managed by ICE without the IAS:

- **Single-Entity Failures:** Certain entities within ICE are crucial for its operation, namely the Supervisor. Faults within the Supervisor and its control algorithms are typically difficult to detect within ICE. Certain faults associated with the processing of patient data cannot be detected by any entity within ICE and therefore fall under the purview of IAS.
- **Multi-Entity Failures:** During the flow of data or commands from apps to the medical device and vice-versa, a component $e1$ in ICE faults along with another component $e2$, where $e2$ is responsible for detecting the fault of $e1$. For example, $e1$ is a medical device which fails to collect a measurement in response to an Supervisor's command, and $e2$ is the Supervisor, which does not detect this despite *not* receiving data from the medical device in the reverse direction.

– **Entity-Link Failures:** During the flow of data or commands between two adjacent components $e1$ and $e2$ that share a communication link, the link faults (i.e., alters, delays, or loses data) and the receiver entity does not detect the fault. For example, the link between the Supervisor and the NC is noisy and results in the altering of the command being sent by the controller (to the medical device), and the NC does not detect this alteration as it does not correctly verify the CRC.
– **Combination Failures:** Finally, combination failures occur when more than one of the above three failures manifest simultaneously. As each of these instances are failures not detectable within ICE, a combination of them will also be undetectable, and would therefore fall under the purview of IAS.

Given the dynamic nature of ICE, many of the requirements may be difficult to design for without making IAS arbitrarily complex. At this stage, however, our goal is not to prune for detectability but to identify potential ICE failure categories that can only be detected by a decoupled entity, such as IAS.

As note of caution, we stress that our goal is to design an alarm system that is as generic and reusable as possible, and one that works for a variety of apps and interoperability situations. Hence, we keep the description of our three hazards as general as possible. It is therefore likely that some of the constituent hazards may not apply fully in specific IMDS instances. For example, consider a cardiac activity monitoring IMDS that collates sensing devices such electrocardiogram, photoplethysmogram, and continuous blood pressure to develop a complete picture of the patient's cardiac process. This IMDS will not harm the patient in any way through incorrect actuation. Therefore several sub-trees of H1 and H2 will not be relevant for this particular IMDS. The alarm system can be configured to ignore any faults reported from such sub-trees at run-time.

4 Case Study

In this section, we present a case study that demonstrates the application of our requirement generation approach to developing the requirements for an alarm system for IMDS. Given the space constraints, we do not provide an extensive fault-tree for the case study, but cover the four types of failures presented earlier. The IMDS chosen for the purposes of this case study is one that facilitates the clinical scenario of safe airway-laser surgery (ALS-IMD) in the context of ICE by providing a safety interlock between the ventilator and the laser. The ventilator supplies oxygen to an intubated patient, and needs to be stopped when the laser is used to make an opening in or near the patient's throat. This has the potential danger of a fire if the laser is activated while high oxygen concentration is supplied by the ventilator. Whenever the laser is being activated, a human operator must block the air path from the oxygen concentrate while ensuring that the patient does not remain without oxygen for too long [10]. In traditional operating room environments, clinicians are supposed to be aware of potential fire and patient hypoxia problems. IMDS for safe airway-laser surgery must meet the following two safety invariants (requirements): (1) **R1:** The supply of oxygen

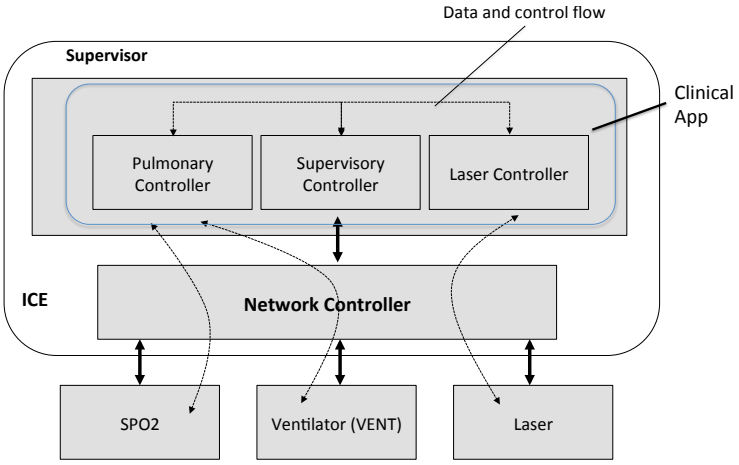

Fig. 5. Airway-laser surgery interoperable medical device system model

from the ventilator must be blocked during the use of the laser to prevent surgical fire; (2) **R2:** The oxygen supply must be resumed within 4.5 minutes or if the patient's SpO2 level falls below 40% [10].

In an ICE context, where safety properties are enforced not by dedicated hardware but by an app running on the supervisor, failure of ICE components to function correctly may jeopardize safety. In order to mitigate such situations, we need an external entity like IAS that can look at the IMDS holistically and detect failures with its operation. The rest of the section is divided into two parts: we begin with a description of the system model ASL-IMD, then discuss how an IAS would be used in an ICE-based implementation of ASL-IMD.

4.1 ALS-IMD System Model

The system model for interoperable medical device system aiding in the safe execution of airway-laser surgery is illustrated in Figure 5. There are three medical devices in this interoperability setup — laser, SpO2, and ventilator. **Individual medical devices may have their own internal logic to detect conditions that affect patient safety**. An ASL app is responsible for making sure that the two safety constraints are satisfied at all times. As in [10], the app makes use of two dedicated "low-level" controllers: a pulmonary controller for the SpO2 and the ventilator, and a therapeutic controller for the laser system. The pulmonary controller maintains a pulmonary model and generates a contingency plan for the ventilator and the laser to follow. These contingency plans provide hard limits for when the laser should stop and the ventilator switched back on in order to ensure R1 and R2 are never violated, providing patient safety even in the presence of network failures. The lower-level controllers are thus closed-loop themselves and will attempt to keep the patient safe even if all else fails. Each of low-level controllers is coordinated by a "high-level" controller within the app

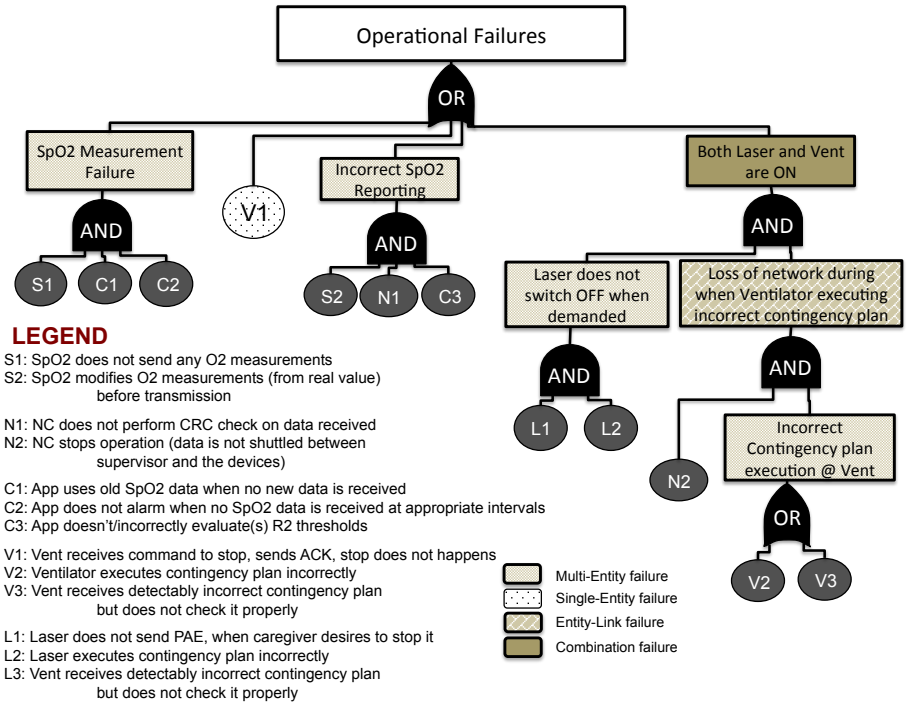

Fig. 6. Airway-laser surgery interoperable operational failures

which determines when to start the laser and stop the ventilator and vice-versa depending upon the needs for the caregiver during the surgery.

Now that we know how ASL-IMD works, we turn to the interoperability hazards of an ICE-based implementation of ASL-IMD. To this end, we build a fault-tree that describes the various failures that can occur with ALS-IMD which cannot be detected by entities in ICE and so require an external/decoupled entity. Figure 6 captures operational faults that can occur within the ALS-IMD system and the resulting failures that arise from these individual faults. The leaf nodes represent individual faults in some entity within ICE, and these faults are combined to create operational failure scenarios.

We now consider these faults from the perspective of the characterization given in Section 3. We can see that all of the identified faults fall within the four categories of faults that form the requirements for the IAS. The shading of nodes in Figure 6 illustrates the categorization of the identified failures. This lets us conclude that an IAS module, built to satisfy these requirements, would successfully raise alarms to notify operators about failures in the ALS-IMD system. As mentioned before, the goal is not to be comprehensive with the fault-trees but to demonstrate how they are generated.

The IAS system will require various fault detection modules that detect the problems specified in the leaf nodes in our fault trees. Many of these detection modules may be IMDS-specific and need to be loaded at run-time when the IMDS is assembled. Our goal with IAS is to build a modular or parametric

alarm system that can incorporate detection modules into a larger whole. An important consideration in integrating these detection modules into IAS is the resultant complexity of the alarm system, which can make any problems with IAS itself difficult to detect. One might even decide to leave out the detection modules for certain faults, thus rendering them undetectable, as long as they do not necessarily cause immediate harm to the patient.

5 Related Work

Integrated Modular Avionics: In aviation, Integrated Modular Avionics (IMA) describes an architecture fairly similar to ICE. The major difference, however, is that components are assembled and integrated beforehand, while, to extend the aviation analogy to ICE, we would be swapping equipment in and out while the aircraft is in flight. IMA describes a distributed real time network of computing platforms known as modules. Modules are networked to each other, to devices of the aircraft such as sensors/actuators and the environment. An IMA module is a layered architecture, described in the avionics standard ARINC 653 [13]. This standard is based on the concept of partitions. Each partition is an area separated from the operating system for scheduling and memory space purposes. Each partition contains an application of the avionics software. Applications have access to common services of the operating system. The operating system interacts with the supporting hardware via a hardware interface layer. Applications can have different levels of criticality and are executed independently. IMA consists of a health monitoring system for fault detection and reporting of application software, OS and hardware failures.

The role of apps in ICE, rather than performing a specific functionality, is to determine the medical devices used in each case and orchestrate their executions. ICE Supervisor is compared to an IMA platform which encapsulates several apps and interacts with the network. Concerning the IMA health monitoring system, it provides fault detection only of the IMA platform. Contrary to IAS that detects faults of the whole architecture, IMA health monitoring systems do not detect faults coming from the network, devices of the aircraft and the environment.

IEC 60601-2-8: The IEC 60601-1-8 standard [7] defines alarms for medical equipment, their associated problems, and provides some suggestions for risk mitigation strategies. Primary considerations are alarm source identification, distractions caused by alarms, and false positive alarms. The system being monitored is best described as a distributed system with associated components, so IAS fulfills the role of a *technical alarm* that ensures, as required by 60601-1-8, that the failure of a component of a distributed system must generate a technical alarm condition. Note that IAS *is not a primary alarm* — these are generated by devices and/or clinical applications. IAS is a secondary alarm which detects component failure when those components themselves either do not detect it or are acting maliciously. The IEC standard was not meant for use in the presence of malicious devices/actors. IAS is compliant with 60601-1-8, and is a step toward making the system being monitored compliant with 60601-1-8, assum-

ing adequate documentation, and an external log, provided by the interoperable system being monitored, for alarm recording.

IEC 80001-2-5: The IEC 80001-2-5 standard [8] deals with *distributed* alarm systems (which are only briefly mentioned in IEC 60601-1-8). These systems handle *multiple medical devices simultaneously*, and their functionality may not be housed within one physical location, and therefore subject to additional safety issues such as transmission delays and lost messages within the medical IT-network. Types of systems include guaranteed delivery, guaranteed delivery with confirmation, and "informational" systems which do not guarantee delivery. Only those systems offering guaranteed delivery with confirmation are sufficiently safe to be used for primary notification. Once again, it is important to recall that IAS is not a primary alarm. Further, defining the means of communicating IAS-generated alarms are beyond the scope and space constraints of this work.

ICE Logging: In [1], the authors discuss the design of a data logging system for an Integrated Clinical Environment (ICE). The data logger is contained in the ICE Network Controller and logs data from all medical devices. Log data is useful for debugging network interactions, clinical event analysis, analyzing patient outcomes and developing advanced clinical algorithms. The data logger provides options to allow user decide the granularity of details of the logs, a suitable clock logic to establish causal ordering between events in cases where real-time clock is not available, special formats for interpreting data between connected devices, security options for ensuring trustworthiness of the log and methodologies for clinical log and debugging playback of the log data. Note that this is both a technical and a clinical data logger.

6 Conclusion

In this work, we develop requirements for a technical Interoperability Alarm System (IAS) for dynamically composable on-demand IMDS. As part of our work we define the scope of IAS as distinct from a physiological or clinical alarm. We use the ICE architecture as the basis of our IMDS for alarm requirements generation. Our approach is to use hazard analysis and fault-trees to systematically and comprehensively categorize the various faults within the ICE architecture which the IAS would be responsible for detecting. The faults thus identified have an important common characteristic — they are all problems that cannot be detected from within ICE by the various ICE entities because they are a result of the simultaneous faults in more than one ICE entity. This work can be generalized to other cyber-physical systems, but is particularly important (and indeed difficult) in the medical system space, where systems are expected to be assembled as needed, with no dedicated integrator or integration testing of a particular system configuration. While the described alarm system cannot detect all failures emergent from interoperability (failures which would not be possible in a non-interoperable system), it significantly increases our assurance that an interoperable medical system will function as intended, and that operators will be notified when it deviates from expected behavior.

As part of the future work, we plan to design the IAS architecture including the required monitors and the alarm logic blocks, develop the specification language and associated vocabulary for event descriptions, and implement IAS as part of an existing ICE implementation.

Acknowledgments

This work was partially funded by NIH grant 1U01EB012470 and NSF grants CNS 1224007, CNS 1239543, and CNS 1253930.

References

1. D. Arney, S. Weininger, S. F. Whitehead, and J. M. Goldman. Supporting medical device adverse event analysis in an interoperable clinical environment: Design of a data logging and playback system. In *ICBO*, 2011.
2. ASTM 2761: Medical devices and medical systems — essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE), 2013.
3. A. Avizienis, J. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1), 2004.
4. M. Clarke, D. Bogia, K. Hassing, L. Steubesand, T. Chan, and D. Ayyagari. Developing a standard for personal health devices based on 11073. In *EMBS*, 2007.
5. J. Hatcliff, A. King, I. Lee, A. Macdonald, A. Fernando, M. Robkin, E. Vasserman, S. Weininger, and J. M. Goldman. Rationale and architecture principles for medical application platforms. In *ICCPS*, 2012.
6. Health Level Seven International. `http://www.hl7.org/`.
7. IEC. Medical electrical equipment – Part 1-8: General requirements for basic safety and essential performance – Collateral Standard: General requirements, tests and guidance for alarm systems in medical electrical equipment and medical electrical systems, 2008.
8. IEC. Application of risk management for IT-networks incorporating medical devices – Part 2-5: Application guidance – Guidance for distributed alarm systems, 2014.
9. Integrating the healthcare enterprise. `http://www.ihe.net/`.
10. W. Kang, P. Wu, M. Rahmaniheris, L. Sha, R. Berlin, and J. Goldman. Towards organ-centric compositional development of safe networked supervisory medical systems. In *CBMS*, 2013.
11. A. King, D. Arney, I. Lee, O. Sokolsky, J. Hatcliff, and S. Procter. Prototyping closed loop physiologic control with the medical device coordination framework. In *SEHC*, 2010.
12. K. Lesh, S. Weininger, J. Goldman, B. Wilson, and G. Himes. Medical device interoperability — assessing the environment. In *HCMDSS-MDPnP*, 2007.
13. P. J. Prisaznuk. ARINC 653 role in integrated modular avionics (IMA). In *DASC*, 2008.
14. A. Tolk, S. Diallo, and C. Turnitsa. Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering. *Journal of Systemics, Cybernetics and Informatics*, 5(5), 2007.