Dependable Integrated Clinical System Architecture with Runtime Verification

Yu Jiang[1], Mingzhe Wang[2], Han Liu[1], Mohammad Hosseini[3], Jiaguang Sun[1]

School of Software, Tsinghua University, China[1]
School of Software, Beijing University of Post and Telecommunication, China[2]
Department of Computer Science, University of Illinois at Urbana-Champaign, USA[3]

***Abstract*—Medical devices are essential for the practice of modern medicine, and the standard open-source integrated clinical environment (`OpenICE`) has been well designed and widely adopted to improve their interoperability. With `OpenICE`, it is easy to connect individual devices into the integrated clinical system to provide a coherent patient care.**

In this paper, we present `ICERV`, the first online verification approach for the `OpenICE`, to ensure the dependability (mainly for the safety and security) of the integrated system and the involved patient and clinician. The key idea is to customize runtime verification technique to provide a transparent verifying infrastructure to continually intercept the communication commands and messages of those devices, based on which, we can formalize the safety and security requirements as past time linear temporal logic expressions for verifier generation and online formal verification. If any requirements violate, predefined warnings or exception handling actions will be triggered timely to prevent hazards and threats. We have implemented and seamlessly integrated the approach without any changes to the source code of `OpenICE` nor the code of the upper-level applications or supervision, and the real device is used for evaluation to demonstrate the effectiveness.

***Index Terms*—Dependability, Runtime Verification, Safety and Security, Medical CPS, Integrated Clinical Environment.**

I. Introduction

Traditionally, clinicians need to continuously monitor and operate all separate devices. A large number of devices and medical information makes the job very stressful and error-prone. In order to reduce the burden on the clinician and avoid the possibility of human errors, the standard open-source integrated clinical environment (`OpenICE`) [15], has been well designed and widely adopted to improve their interconnectedness. `OpenICE` is a distributed system platform for connecting network nodes (e.g., medical devices, decision support system) together, and supports the open standard ASTM F2761 [1], With `OpenICE`, it is easy to connect individual medical device with network interfaces into the integrated clinical system through the network to obtain comprehensive data to personalize care.

With the wide adoption of `OpenICE`, its dependability is more and more important. Potential hazards will lead to irreversible dangerous scenarios of the patient (e.g., mistakenly or viciously sending a command to emit the laser when the oxygen ventilator is still working will trigger fire on the trachea of the patient). Unlikely, such hazards are common during the medical practice and have been documented in many kinds of clinical literature. For example, an emergency may disperse the attention of the clinician, then some information presented on the supervisor of `OpenICE` may be ignored and some operation commands to the medical device may be sent by mistake. Also, information produced by the connected medical devices and commands issued by the supervisor may be lost and distorted due to network failure or attack. Hence, it is highly needed to rigorously ensure the dependability and prevent potential safety hazards and security threats, which is challenging due to the dynamic and distributed features of `OpenICE`, and the increasing number and complexity of the integrated medical devices.

More specially, it is not easy to verify those hazards and threats in advance because of the dynamic feature that medical devices can plug-and-play and leave at any time, and the patient reactions to treatment of medical devices are uncertain and irreversible. For example, when the laser scalpel is registered in the system, additional on fire safety hazards with existing ventilator should be dynamically added and verified. The distributed feature of `OpenICE` that any medical device can send arbitrary information to supervisor increases the difficulty of statically analysis, and can be easily abused by attackers. For example, the command to stop the ventilator before emitting the laser may be distorted easily.

Proposed Approach: We present an online runtime verification approach `ICERV`, to prevent the safety hazards and security threats of the `OpenICE` . The key idea is to provide a transparent runtime verifying infrastructure to continually intercept the communication commands and messages among those integrated devices and applications, based on which, safety requirements and security requirements are formalized as past time linear temporal logic expressions and further automatically implemented as verifiers. Those verifiers are lightweight and transparent, which can be integrated without changes to the `OpenICE` and the application and device adaptors integrated on `OpenICE`. Real devices and applications based evaluation are conducted to test the efficiency, the false negative rate for detection of typical safety hazards is about 20%-30% better than the clinical environment with traditional decision support systems.

II. Clinical Environment

Open-source integrated clinical environment (`OpenICE`) is an open clinical platform for medical device and applications integration. It implements the standard ASTM 2761-09, which regulates the essential principles of safe and reliable integration equipment. More specifically, `OpenICE` is a distributed system platform for connecting network nodes: (1) medical devices, (2) recording applications, (3) clinical decision support systems, and (4) other health care systems. The current version of `OpenICE` enables users to convert heterogeneous medical device data from supported devices

978-1-5386-3093-8/17/$31.00 ©2017 IEEE

into a common structure and protocol, and to exchange that data with clinical applications on a different machine. Furthermore, OpenICE automates peer-to-peer node discovery, data publishing and subscribing between nodes, as well as proprietary medical device protocol translation. Instead of stand-alone devices that can be certified, and used to treat patients alone, integrated medical clinical systems based on OpenICE are increasingly used in hospitals.

The overall structure of OpenICE is demonstrated in the figure of the appendix, from which, we can see that it has two main components: OpenICE Supervisor and OpenICE Device-Adapter. The device-adapter software acts as a bridge that will connect both real and simulated medical devices into the network. It translates the proprietary device communication protocols into the standard OpenICE data structures and communication protocol. The supervisor software runs clinical applications and subscribes to all medical devices.

III. Dependability of OpenICE

For the dependability of OpenICE, we mainly address the safety and security, others aspects of dependability such as reliability are not addressed within this framework.

Security Threats and Requirements Extraction: Security of the general system can be defined as the ability to ensure that both data and operational capabilities can only be accessed when authorized [16]. The need for security in OpenICE based medical clinical system is manifold because of the following features: (1) It is mission critical and any security compromise of either the supervision or the integrated medical devices can have profound consequences to the involved patient. A case in point is the attack on ventilators that force it to open when the laser is emitted, which will lead to fire disaster of the patient. (2) The patient information is sensitivity and privacy, and it can be easily exploited by malicious entities leading to abuse, discrimination, and even misdiagnosis. Examples include intercepting and distorting the psychiatric records, genetic information, and real-time vital signs.

Based on the OpenICE security models [4] and the above features of OpenICE based medical clinical system, some of the most common security threats and requirements are chosen and classified as below:

1. *Threats of Application Tracking.* The requirement for the prevention of registration of unauthorized applications, which disturb the functions of the integrated system. For example, an application to deliberate generation of false alarms or suppression of real alarms raised by the system in case of emergencies.
2. *Threats of Device Tracking.* The requirement for the prevention of registration of unauthorized medical devices, the unauthorized fadeout of existing medical devices, virtual backup copy of existing medical devices and unauthorized actuation of existing medical devices, which destroy the functions of the integrated system. For example, fadeout of an existing medical device and then create a virtual backup copy wold lead to system crash and operation failure.

Also, there are some other general security requirements for health information system such as requirements for implantable medical devices such as the device setting security that the system should enable authorized update of the software. They can be addressed by the existing work [19], and we mainly focus on the two categorizations and address them with the proposed approach.

Safety Hazards and Requirements Extraction: The safety requirements of OpenICE based medical clinical system can be defined as the avoidance of hazards due to the operation of medical devices and applications under fault conditions [2]. It is more important in OpenICE because (1) The medical device is directly working on the real patient, and hazards of the device may result in failure of the treatment and even leading to patient death. (2) The integration of multiple devices through the network enables parallel automatic or semi-automatic operation, which leads to more complex hazards combination and error-prone situation. Besides, safety requirements are highly application dependent. Based on the OpenICE safety rules of the standard ASTM F2761 and standard ISO 60601 [18], accompanied with some hazard analysis of real OpenICE based applications, some of the most common safety hazards and requirements are chosen and classified as below:

1. *Hazards among different medical devices.* The requirement for the prevention of dangerous combination status of integrated medical devices. For example, both ventilator and laser scalpel should not be in-operation at the same time, if not, a surgical fire would be caused. Another case is that the electromagnetic interaction of the headphone with the patients' body gets coupled with the electromagnetism induced by the pacemaker on the patients' heart and deactivates it.
2. *Hazards from device and application to patient.* The requirement for the prevention of dangerous operations of the medical device on patients. For example, the ventilator should only remain at in-operation status after surgery, if not, the patient will have brain damage due to hypoxia. Another example is that the insulin infusion pump should stop operation immediately once the patient is detected in severe hypoglycemia.

From the categorizations, we can see that different clinical applications and medical care practices will lead to very different safety requirements. Also, there are some other safety requirements from the patient to device that the tissue growth around the implanted sensors can hamper sensing and communication capabilities, and some safety requirements from the supervision to the device that the communication is broken and the command is lost. These kinds of safety hazards can be addressed by improving the quality of the medical devices or network, and we will mainly focus on the two categorizations above.

IV. Online Verification of OpenICE

The architectural overview of the proposed online verification approach is presented in Figure 1. The main difference from the original OpenICE is the additional component verifier supervisor and safety verifier, which act as both a secure and functional layer for protecting the original components OpenICE supervisor, application, and device. Then, all devices and applications' communication with the OpenICE supervisor can be intercepted by the additional layers, and thus the desirable safety and security

policies are enforced through the verification on the message. Also, the additional layer does not require any change to OpenICE. The only requirement is to configure the supervisor verifier to listen at the standard port and the OpenICE supervisor to listen at the hidden port. In this way, all the applications and device interface in the system remain the same (sending requests to the default port, sending and receiving messages), and are not even aware of being verified.

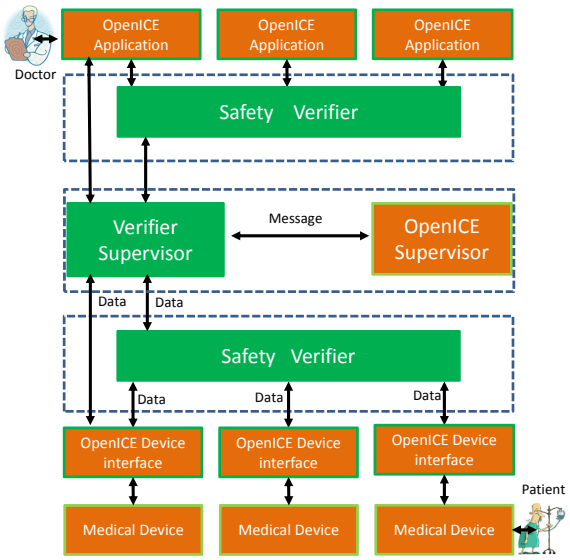

Fig. 1. Architecture of the proposed online verification approach, where we add the verifier layer into the original OpenICE architecture.

Verifier Supervisor: We have designed an IP address-based device access control specification and an identifier-based application access control specification that allows the security policies to be enforced within verifier supervisor. For example, access control policies such as allowing certain applications to request messages or send commands of certain types or in certain situations can be enforced by checking the application identifier in the request, and the verifier supervisor rejects the communication establishment if the policy is violated. Besides the secure verification of access and communication, the verifier supervisor also takes the responsibility of generating the safety verifier dynamically according to the changing configurations and situations of the environment. They are generated from the safety requirements and implemented as virtual applications that can receive and send messages, and verify the property originated from the requirement. The generated verifier contains four java components: *message parser, event generator, property verifier,* and *response action*.

Safety Verifier: There are two types of safety verifier. If the safety requirement is not related to the communication between two devices or two applications, the generated safety verifier does not need to handle the communication, and just intercepts the transmitted message between the device interface (or application) and the OpenICE Supervisor, and verify the related safety property. Another type is for the safety requirement that is related to the communication between two devices or two applications. In this case, the verifier supervisor needs to keep track of all communication connections of the devices and applications, and manipulates the point-to-point communication addresses so that the generated safety verifier acts as a transparent inspector between the communicated applications or devices. When the verification detects some dangerous situations, timely actions would be adopted to ensure safety. For example, when the OpenICE Supervisor sends a command to the laser scalpel to emit laser, the safety verifier would check the status of the ventilator. If the value of the device status is true, the conflict event would be true, the property is violated, and the violation action that drops the command and sends warning would be taken immediately.

A. *Safety Requirements Formalization*

Based on the extraction of safety requirements presented in Section III, we now show how to describe and formalize the requirements into a verifiable property based on the message, event, and action mentioned above. Details of the input format are presented in the appendix and we use a simple example presented in Figure.2 to demonstrate the formalization.

```
Safety Specification: TypeOne: laserOnFire
{
        bool statusVentilator;
        bool commandLaserScalpel;
        float statusO2;

        event onLaserScalpel = commandLaserScalpel == True;
        event safeSurgery = (commandLaserScalpel == True
                && StatusVentilator == False && statusO2 < fireThreshold);
        event dangerousSurgery = (onLaserScalpel
                && ( StatusVentilator == True || statusO2 >= fireThreshold));

        property patientSafety = [ ] (safeSurgery /\ ¬dangerousSurgery);
        @violation{
                System.out.println("property violated!");
                commandLaserScalpel = false;
        }

}
```

Fig. 2. Safety requirement specification for laser surgery on-fire.

This is a safety requirement specification for the safety verifier of type one, where the laser scalpel should not be operated when the ventilator is in operation. If not, the surgical fire would be caused. There are three variables defined in this specification, the status of the ventilator, the command operation of the laser scalpel, and the corresponding oxygen level. Their values are available in the OpenICE supervisor and could be abstracted by the component *message parser* from the configured communication interface. Based on these variables, we define three events, onLaserScalpel, safeSurgery, and dangerousSurgery. The simple event can be defined as boolean evaluations of variables, and complex event such as dangerousSurgery can be defined on both simple event and variables. For example, an complex event dangerousSurgery will be triggered when an onLaserScalpel event happens and the status of the ventilator is on or the oxygen level crosses the threshold at the same time. The definition and the evaluation of event are implemented in the component *event generator*.

Then, we can specify safety properties based on those events. The property patientSafety means that under any condition, when the event onLaserScalpel happens, it must

also be a safe surgery indicated by the event safeSurgery. We make use of logic plugins of monitoring oriented programming(MOP [3]) to specify temporal properties over events, such as finite state machine, linear temporal logics, and past time linear temporal logics(*ptLTL*), etc. Furthermore, predefined additional actions and warnings can be adopted in case of property violation, as implemented in the component *response action*. For example, in the laser surgery case, when the property of patientSafety is not violated, we would revise the value of the commandLaserScalpel to be false, and the revised value will be sent to the device to avoid the hazard of laser surgery on-fire.

B. Security Requirements Configuration

Based on the extraction of security requirements presented in Section III, we now show how to describe and enforce the access security policies to acquire a secure device tracking and application tracking. The tracking policies are currently described into two segments: sending policy and receiving policy, where each access rule is written as a key with an assignment to the key.

1. *Sending access rule.* the key means the message or command type, and the assignment includes the identifier that is allowed to send the message.
2. *Receiving access rule.* the key means the message or command type, and the assignment includes the identifier that is allowed to receive the message.

The identifier is at a host level, with IP address accompanied with the optional device name or application name. Because there are multiple devices, IP address group is also supported in the specification. We use the example presented in Figure 3 to illustrate the configuration.

```
Security Specification: medicalDeviceAccess
{
        IP address   ventilator  = 73.45.141.184;
        IP address   laserScalpel  = 73.45.141.185;
        IP address   supervisor = 127.0.0.1;
        IP group     medicalDevice = ventilator, laserScalpel;

        sending      getDeviceStatus = supervisor;
        @violation {
                      System.out.println("Security threats!");
                     }
        sending      removeDevice = supervisor;
        sending      sendDeviceStatus = medicalDevice;
        receiving    onVentilator = ventilator;
}
```

Fig. 3. Security requirement specification for access of medical device.

In the security requirement specification, the IP address aliases are defined for the ventilator, laser scalpel, and supervisor. The IP group medicalDevice contains the first two IP addresses. Then, sending and receiving access rules are defined. The first rule means that the message type of getDeviceStatus can only be issued by the supervisor. If not, security threats would be reported, as defined in the violation action. The second rule means that an existing medical device can only be removed by the supervisor. The last rule means that the message type of onVentilator can only be accepted by the ventilator. For example, if the X-Ray Ventilator Sync application tries to send the message type of onVentilator to the laserScalpel, threats may be reported. All rules are implemented in the `verifier supervisor`, to decide whether a communication request should be handled by the `OpenICE supervisor` or not.

V. Evaluation

We have implemented and integrated the approach with `OpenICE`, and experimented them on popular clinical system integration scenarios with injecting typical safety hazards and security threats. We include the comparisons with traditional decision support systems such as Spock [21]. The intentionally injected 42 safety hazards [11], [14] and 13 security threats [4] can all be detected and mitigated well with `ICERV` approach, while only 12 safety hazards can be detected by Spock. This section presents the evaluation and answers the following two questions.

Q1. Can `ICERV` approach effectively detect and avoid real-time safety hazards of `OpenICE` ?

Q2. Can `ICERV` approach effectively detect and avoid real-time security threats of `OpenICE` ?

A. Experimental Setup

Implementation and integration. We setup a full-featured `OpenICE` integrated clinical system with the help of Carle hospital. In the integration, the following hardware should be used: (1) Desktop or laptop computer running device-adapter software, (2) Desktop or laptop computer running the OpenICE Supervisor with `ICERV` and Spock, and (3) WiFi router connecting every device.

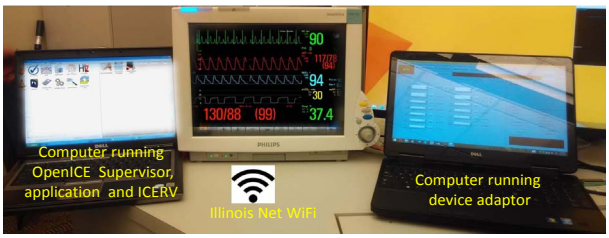

Fig. 4. Integration of `ICERV` and `OpenICE` based medical CPS.

The `OpenICE` system's discovery mechanism and pub-/sub functionality allow us to run a device simulator anywhere within the network. This simulator will be visible to any `OpenICE supervisor` on the same local network segment, and we can use the simulator to simulate the medical device such as infusion pump, in case of the situation that real devices are not available. Currently, we have access to a real device of ventilator Drger Evita 4 and Philips MP70 Intellivue, and other devices are configured as a simulator in our experiments.

Safety hazards and security threats. Based on the hardware platform, we initialize the safety hazards and security threats related to two existing applications within `OpenICE`, the first is for the application named infusion safety and the second is for the application named X-Ray ventilator sync. These two applications are chosen because they are widely used. For example,the infusion pump is adopted in delivering medicine such as antibiotics or pain relievers, and the ventilator is adopted in most airway laser surgery. Also, there are many existing safety hazards [11], [12], [14] and security threats [4] documented on the two applications, which can be used for injection directly.

B. Results and Statics

Safety hazards. Let us see the evaluation on the safety hazards first. In the table I, we present the results of the 42 safety hazards described in [11], [14]. As the table shows, the ICERV approach detects 39 out of the 42 intentionally injected safety hazards and can take the corresponding response actions to avoid further disaster. Besides, there are three specifications not being detected because of the lack of additional information, leading to the result of the *safety verifier* unknown. The Spock can only detect 12 hazards of the type that value crosses the threshold.

TABLE I
SAFETY HAZARDS INTENTIONALLY INJECTED AND DETECTED

Property	Injected	ICERV	Spock	Time
Infusion Pump	28	26	8	6.8(ms)
Ventilator	14	13	4	7.6(ms)

An example for the infusion pump is demonstrated as an illustrative evidence. For a stroke patient, rt-PA is delivered through the infusion pump. During the infusion procedure, if blood pressure and blood glucose level cannot be controlled under acceptable ranges ((BP>180——BP<90)&& (HR>120——HR<60)), or signs of brain hemorrhage appear, the stroke team should stop the rt-PA infusion. Timely response based on the online verification is highly desirable, because every second the huge number of brain cells die, for example, 32000 brain cells will die within every second a clot blocks blood flow to the brain. We inject the hazard as below: (1) set the BP and HR through the patient monitoring under control, and infusion safety application keeps driving the simulator of the infusion pump, (2) intentionally set the BP and HR to cross the threshold, and still keeps the infusion safety application driving the simulator. With ICERV approach, the command sending to the simulated infusion pump will be revised in an average of 6.8 milliseconds to avoid the hazards.

For the ventilator, we can also inject similar hazards as below: (1) set the blood oxygen level to low level, and the ventilator is off. (2) the sync ventilator application opens the ventilator Drger Evita 4. With Spock, the ventilator will be opened immediately. While with ICERV approach, the status of laser scalpel will be acquired for verification first, and the response action would change the command to stop in case of laser emitting. Others can be injected in a similar way, and the performance keeps the same. Based on the results, the answer to the question **Q1** is in affirmative.

Security threats. Let us see the evaluation on the security threats. In table II, we present the results of the 13 security threats [4], initialized according to the description of section IV. As the table shows, the ICERV detects 13 out of the 13 intentionally injected security threats and takes the corresponding actions to avoid further violations, while the traditional decision support systems have no support.

TABLE II
SECURITY THREATS INTENTIONALLY INJECTED AND DETECTED

Property	Injected	ICERV	Spock	Time
Infusion Pump	5	5	0	2.8(ms)
Ventilator Sync	6	6	0	3.1(ms)
HL7 Exporter	2	2	0	1.7(ms)

A simple example about the security threats injection for Health Level Seven International (HL7) application is illustrated as follow. The HL7 Exporter application exports data to Health Level Seven International database. The host field and port number are the net address of the HL7 export. Other address or other applications would be not allowed. Then we can intentionally change the address of the application. The access will be forbidden within 1.7 milliseconds. Based on the result, we can also respond to the question **Q2** positively.

Furthermore, we do some real verifications in SimMan laboratory for evaluation. We use SimMan patient simulator and some virtual device adapter to set the vital signs of the SimMan and the value of real-time data monitor device Phillips IntelliVue MP70, which can be furthered passed to the OpenICE supervisor, application and the generated ICERV runtime verifier.

TABLE III
DETECTED WARNING COMPARISONS FOR DIFFERENT SCENARIOS

Property	Injected	Spock	ICERV	Time
UN_HR_BP	1000	925	997	1.4(ms)
HR_BP_20	1000	729	998	1.8(ms)
HR_Bp_30	1000	685	997	2.0(ms)
Security_BP	1000	0	994	1.1(ms)

We initialize four typical requirements, the first is the infusion pump safety hazards about blood pressure and blood glucose level violations mentioned above, the second and third are the temporal extension for continues 20 second and 30 second violation respectively, and the fourth is the data exportation security. The second column of the table III is the number of violations that we insert into the virtual patient through the SimMan patient simulator, the third column is the violations manually detected by staring at Spock and OpenICE supervisor, and the fourth column is the violations detected with the enhancement of ICERV. From the third column, we find that the accuracy of nurse decreases along with the complexity of the hazards. But for the online verifier, it performs steadily, and the false negative rate for safety hazards detection is about 20%-30% better than the clinical system with only human inspection of the traditional decision support system Spock, which has no support for security attack detection neither. During analysis of the log, we found that the ICERV approach will produce 1000 warnings, but two or three may still be ignored by the nurse because of noise that disturb them. According to the simulation in the SimMan laboratory, it is reasonable to draw the conclusion that the online verification helps produce a safe and secure integrated clinical environment.

VI. DISCUSSION

Validity Discussion: Currently, there are three limitations. The first is for the expression ability of the safety requirement specification and security requirement specification. Some complex requirements are not supported, and the corresponding safety verifier cannot be generated. We mainly address those hazards and threats abstracted in section III. The second is for the data interception ability. For the safety verifier of some requirements, we can not get the data automatically. For example, the verification for a ventilator related property needs the value

of oxygen level, which is not stored in OpenICE. One possible way to overcome this limitation is to add more sensors into the integration. The last but most important is the network situation and attack. We assume the network in the clinical environment is small, stable so that network delay is negligible, and leave an in-depth network attack model with all various parameters as our future work.

Lessons and Related Work Discussion: Traditionally, researchers have focused on bypassing and transforming the safety assurance problem of medical system into a well understood problem of formal model reachability analysis in traditional embedded system design [13], [7], [10], [9], [20]. For example, in works such as [11], [14], infusion pump software and pacemaker have been modeled as a timed automaton, and several static assumptions and abstractions are made out the dynamic nature of the physical environment. In works such as [17], laser scalpel and ventilator are modeled by hybrid automata. These models can be used for off-line verification on safety hazards. Some work about the online verification of medical systems is also presented [8], [6]. For example, and [8], [6] focus on improving the medical best practice without considering the clinical environment and security. **Different from existing research on the model level or just practice monitoring, we address the problem online in deployed clinical environment.**

Balancing security, privacy, and utility is also a necessity in the medical system, and most existing works concern implantable medical devices and body area networks. Huge efforts have been paid to ensuring the security and privacy of the telemetry interface, software, and sensor interface layers of them [5]. Little attention has been paid to the security of the integrated clinical environment, which is also very important as pointed out in [4], where the authors intend to inform future work on building such a comprehensive protocol stack or standardizing protocols and protocol suites for secure next-generation device coordination. **We try to highlight the secure problem of OpenICE and only address part of it in a lightweight manner at the host level without any change to the protocol.**

VII. Conclusion

In this paper, we present the first online verification approach to ensure the safety and security of integrated clinical environment. The key idea is to provide a transparent runtime verifying infrastructure to continually intercept the communication commands and messages of those devices, based on which, we can formalize the safety and security requirements as past time linear temporal logic expressions for verifier generation and online formal verification, and the empirical results demonstrate its effectiveness. The approach and the evaluation provide a guidance for the future safety and security assured design of dedicated integrated clinical environment for better health care.

Acknowledgement

We thank Lui Sha from UIUC for his valuable suggestions, and Binghua Wan from Tsinghua First hospital for his valuable experiment devices. This work is supported in part by NSF CNS 1545008 and NSF CNS 1545002 and Jiangxi Province Major Project 20171ACE50025.

References

[1] ASTMF. Medical devices and medical systems essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ice) part 1: General requirements and conceptual model. 2009.

[2] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta. Ensuring safety, security, and sustainability of mission-critical cyber–physical systems. *Proceedings of the IEEE*, 100(1):283–299, 2012.

[3] F. Chen and G. Roşu. Java-mop: A monitoring oriented programming environment for java. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 546–550. Springer, 2005.

[4] D. Foo Kune, K. Venkatasubramanian, E. Vasserman, I. Lee, and Y. Kim. Toward a safe integrated clinical environment: a communication security perspective. In *Proceedings of the 2012 ACM workshop on Medical communication systems*, pages 7–12. ACM, 2012.

[5] D. Halperin, T. Kohno, T. S. Heydt-Benjamin, K. Fu, and W. H. Maisel. Security and privacy for implantable medical devices. *Pervasive Computing, IEEE*, 7(1):30–39, 2008.

[6] Y. Jiang, H. Liu, H. Kong, R. Wang, M. Hosseini, J. Sun, and L. Sha. Use runtime verification to improve the quality of medical care practice. In *2016 38th ACM International Conference on Software Engineering(ICSE)*. ACM, 2016.

[7] Y. Jiang, H. Liu, H. Song, H. Kong, M. Gu, J. Sun, and L. Sha. Safety-assured formal model-driven design of the multifunction vehicle bus controller. In *FM 2016: Formal Methods: 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings 21*, pages 757–763. Springer, 2016.

[8] Y. Jiang, H. Song, R. Wang, M. Gu, J. Sun, and L. Sha. Data-centered runtime verification of wireless medical cyber-physical system. *IEEE Transactions on Industrial Informatics*, 2016.

[9] Y. Jiang, Y. Yang, H. Liu, H. Kong, M. Gu, J. Sun, and L. Sha. From stateflow simulation to verified implementation: A verification approach and a real-time train controller design. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2016 IEEE*, pages 1–11. IEEE, 2016.

[10] Y. Jiang, H. Zhang, Z. Li, Y. Deng, X. Song, M. Gu, and J. Sun. Design and optimization of multiclocked embedded systems using formal techniques. *IEEE transactions on industrial electronics*, 62(2):1270–1278, 2015.

[11] 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 *Proceedings of the ninth ACM international conference on Embedded software*, pages 155–164. ACM, 2011.

[12] T. Li, F. Tan, Q. Wang, L. Bu, J.-n. 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 *Cyber-Physical Systems (ICCPS), 2012 IEEE/ACM Third International Conference on*, pages 13–22. IEEE, 2012.

[13] A. Y.-Z. Ou, Y. Jiang, P.-L. Wu, L. Sha, and R. B. Berlin. Preventable medical errors driven modeling of medical best practice guidance systems. *Journal of medical systems*, 41(1):9, 2017.

[14] M. Pajic, R. Mangharam, O. Sokolsky, D. Arney, J. Goldman, and I. Lee. Model-driven safety analysis of closed-loop medical systems. *Industrial Informatics, IEEE Transactions on*, 10(1):3–16, 2014.

[15] J. Plourde, D. Arney, and J. M. Goldman. Openice: An open, interoperable platform for medical cyber-physical systems. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, pages 221–221. IEEE, 2014.

[16] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.

[17] F. Tan, Y. Wang, Q. Wang, L. Bu, and N. Suri. A lease based hybrid design pattern for proper-temporal-embedding of wireless cps interlocking. *Parallel and Distributed Systems, IEEE Transactions on*, 26(10):2630–2642, 2015.

[18] A. Turnbull. The use of iec 60601-1 in supporting approvals of medical electrical devices and the role of the new collateral standard iec 60601-1-9, 2007.

[19] K. Venkatasubramanian and S. K. Gupta. Security solutions for pervasive healthcare. *Security in Distributed, Grid, Mobile, and Pervasive Computing*, page 349, 2007.

[20] Y. Yang, Y. Jiang, M. Gu, and J. Sun. Verifying simulink stateflow model: timed automata approach. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, pages 852–857. ACM, 2016.

[21] O. Young and Y. Shahar. The spock system: developing a runtime application engine for hybrid-asbru guidelines. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 166–170. Springer, 2005.