INVITED PAPER

Challenges and Research Directions in Medical Cyber–Physical Systems

A broad overview of emerging applications for these systems is provided in this paper; challenges, promising solutions, and open problems are presented.

By Insup Lee, *Fellow IEEE*, Oleg Sokolsky, *Member IEEE*, Sanjian Chen, *Student Member IEEE*, John Hatcliff, Eunkyoung Jee, *Member IEEE*, BaekGyu Kim, Andrew King, *Student Member IEEE*, Margaret Mullen-Fortino, Soojin Park, Alexander Roederer, and Krishna K. Venkatasubramanian, *Member IEEE*

ABSTRACT | Medical cyber–physical systems (MCPS) are life-critical, context-aware, networked systems of medical devices. These systems are increasingly used in hospitals to provide high-quality continuous care for patients. The need to design complex MCPS that are both safe and effective has presented numerous challenges, including achieving high assurance in system software, intoperability, context-aware intelligence, autonomy, security and privacy, and device certifiability. In this paper, we discuss these challenges in developing MCPS, some of our work in addressing them, and several open research issues.

KEYWORDS | Closed-loop physiological control; cyber–physical systems (CPSs); design challenges; medical device systems; model-based design

Manuscript received July 1, 2011; accepted August 10, 2011. Date of publication October 18, 2011; date of current version December 21, 2011. This work was supported in part by the National Science Foundation (NSF) under Grants CNS-0834524, CNS-0930647, and CNS-1035715 and by the National Institutes of Health (NIH) under Grant 1U01EB012470-01.

I. Lee, **O. Sokolsky**, **S. Chen**, **B. Kim**, **A. King**, **A. Roederer**, and **K. K. Venkatasubramanian** are with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: lee@cis.upenn.edu; sokolsky@cis.upenn.edu; sanjian@cis.upenn.edu; baekgyu@cis.upenn.edu; kingand@cis.upenn.edu; roederer@cis.upenn.edu; vrkis@cis.upenn.edu).

J. Hatcliff is with the Computing and Information Sciences Department, Kansas State University, Manhattan, KS 66502 USA (e-mail: hatcliff@cis.ksu.edu).

E. Jee is with the Computer Science Department, KAIST, Daejeon 305-701, Korea (e-mail: ekjee@se.kaist.ac.kr).

M. Mullen-Fortino and **S. Park** are with the Hospital of the University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: margaret.fortino-mullen@uphs.upenn.edu; soojin.park@uphs.upenn.edu).

Digital Object Identifier: 10.1109/JPROC.2011.2165270

I. INTRODUCTION

The medical device industry is undergoing a rapid transformation, embracing the potential of embedded software and network connectivity. Instead of standalone devices that can be designed, certified, and used independently of each other to treat patients, we will be faced in the near future with distributed systems that simultaneously monitor and control multiple aspects of the patient's physiology. The combination of embedded software controlling the devices, networking capabilities, and complicated physical dynamics exhibited by patient bodies makes modern medical device systems a distinct class of cyber–physical systems (CPSs). We refer to these as medical cyber–physical systems (MCPSs).

MCPSs, due to their increased size and complexity relative to traditional medical systems, present numerous developmental challenges. The long-term viability of MCPSs requires addressing these challenges through the development of new design, composition, verification, and validation techniques. These present new opportunities for researchers in MCPSs and in general embedded and CPSs. For MCPS, we also believe new regulatory procedures to approve their use for treating patients will be needed. The traditional process-based regulatory regime used by the U.S. Food and Drug Administration (FDA) to approve medical devices is becoming too lengthy and prohibitively expensive with the increased MCPS complexity, and we present possible solutions to ease this process.

In this paper, we describe some of the research directions that we are taking toward addressing some of the

challenges involved in building MCPS. The ultimate goal is to develop foundations and techniques for building safe and effective MCPS.

Overall, we advocate a systematic analysis and design of MCPS for handling their inherent complexity. Consequently, model-based design techniques should play a larger role in MCPS design. Models should include devices and the communications between them, but also, of equal importance, the relationship of these devices to patients and caregivers. The use of models will allow developers to assess system properties early in the development process and build confidence in the system design, before the system is built. Analysis of system safety and effectiveness performed at the modeling level needs to be complemented by generative implementation techniques that preserve properties of the model in the implementation. Results of model analysis, combined with the guarantees of the generation process, can form the basis for evidence-based regulatory approval.

The paper is organized as follows. Section II provides a conceptual view of MCPSs and their principal challenges. Sections III–VII present our work in addressing some of these challenges. Section VIII presents a short discussion on some of the open research issues associated with MCPS and Section IX concludes the paper.

II. AN OVERVIEW OF MCPS

MCPSs are *safety-critical*, *interconnected*, *intelligent* systems of medical devices. Traditional clinical scenarios can be viewed as closed-loop systems where caregivers are the controllers, medical devices act as sensors and actuators, and patients are the "plants." MCPSs alter this view by introducing additional computational entities that aid the caregiver in controlling the "plant." Fig. 1 shows the conceptual overview of MCPSs. The devices used in MCPSs can be categorized into two large groups based on their primary functionality: monitoring devices, such as heart-rate and oxygen-level monitors and sensors, which provide information about the patient's physiologic state; and delivery devices, such as infusion pumps and ventilators, which actuate therapy capable of changing the patient's physiologic state. In MCPSs, the monitoring devices can feed data collected to a decision support or administrative support entities, each of which serves a different, albeit complementary, purpose.

Administrative entities, such as electronic health records (EHRs) and pharmacy stores, manage patient health and treatment information collected over a period of time. Given their access to a wealth of personalized information, they have the potential to provide targeted actuation of treatment based on a more holistic view of the patient's health (e.g., by considering potential drug interactions or by taking into account the longitudinal evolution of a specific patient physiological parameter). In this regard, they can assist in fulfilling the need for the continuous care of the patient. Continuous data gathering and management is essential for many of today's health issues such as dealing with the aging population and the rapid rise in the number of people with chronic conditions such as asthma and diabetes.

Decision support entities can process the data collected and generate alarms for many medical emergencies. Alarms are necessary to allow clinicians to know when the patient's

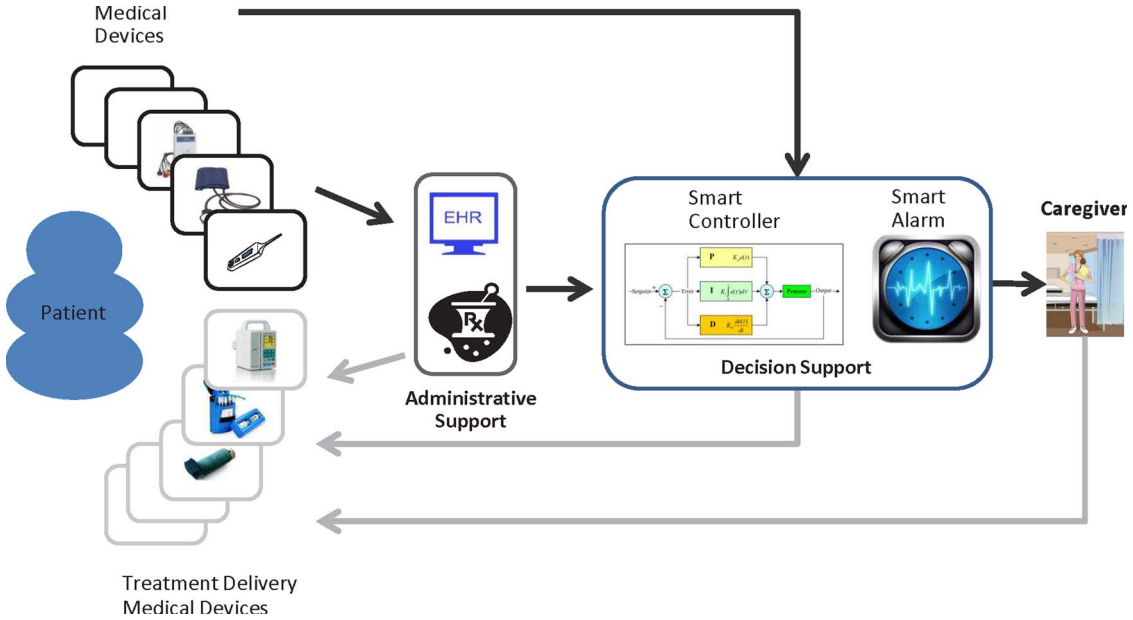


Fig. 1. ***Medical cyber–physical systems: Conceptual overview.***

state has deteriorated and what information is relevant to treat them. However, it is now clear that we must develop smart alarm systems that go beyond the current threshold based methods to provide more accurate, targeted alarms, along with context information about them. Caregivers can analyze that information and can use delivery devices to initiate treatment, thus bringing the caregiver into the control loop around the patient. Alternatively, the decision support entities can utilize a smart controller to analyze the data received from the monitoring devices, estimate the state of the patient's health, and automatically initiate treatment (e.g., drug infusion) by issuing commands to delivery devices, thereby closing the loop.

MCPS Challenges. Building these sorts of MCPS requires addressing several important challenges.

- *High assurance software*: Software plays an increasingly important role in medical devices. Many functions traditionally implemented in hardware—including safety interlocks—are now being implemented in software. Thus, high-confidence software development is critical to assure the safety and effectiveness of MCPS.
- *Interoperability*: As medical devices get communication interfaces, it is essential to ensure that the integrated medical devices are safe, effective, secure, and can eventually be certified as such.
- *Context awareness*: Patient information exchanged during device interoperation can not only provide a better understanding of the general health of the patient, but also enable early detection of ailments and generation of effective alarms in the event of emergencies. Given the complexity of the human body and variations of physiological parameters over patient population, developing such computational intelligence is a nontrivial task.
- *Autonomy*: The computational intelligence that MCPS possess can be used for increasing the autonomy of the system by enabling actuation of therapies based on the patient's current health state. Closing the loop in this manner must be done safely and effectively.
- *Security and privacy*: Medical data collected and managed by MCPSs is very critical. Unauthorized access or tampering with this information can have severe consequences to the patient in the form of privacy loss, discrimination, abuse, and physical harm. Preserving the security of MCPSs is thus crucial.
- *Certifiability*: The complex and safety-critical nature of MCPSs requires a cost-effective way to demonstrate medical device software dependability. Certification of medical devices provides a way of achieving this goal. Certifiability is therefore an essential requirement for the eventual viability of MCPSs and an important challenge to be addressed.

In the next several sections, we briefly highlight some of our current work in addressing the various of the challenges in building MCPSs. We begin with a model-based development for high assurance medical devices in Section III. We then present our work on interconnecting several of these high-assurance medical devices, enabling them to interoperate in Section IV. The data collected from such interoperating devices can be processed to enable the understanding of and informing caregivers of the patient context. We describe several smart alarm systems that can generate effective alarms based on processing multiple streams of patient vital signs in Section V. The availability of context awareness enables autonomous medical systems. In this regard, building MCPSs for safe closed-loop control for patient care delivery is described in Section VI. Finally, we present our work on issues involved with certification of MCPSs in Section VII.

III. HIGH-CONFIDENCE DEVELOPMENT OF MCPSs

Software plays an increasingly important role in the development of new MCPSs. Most new device functionality is software based, and many functions traditionally implemented in hardware—including safety interlocks—are being switched to software. Thus, high-confidence software development is very important for the safety and effectiveness of MCPSs.

Model-based development has emerged as a means of improving software quality. The model-based approach allows developers to perform rigorous model verification with respect to safety and functional requirements, and then through systematic code generation techniques derive code that preserves the verified properties of the model. Such a development process allows one to detect problems with the design and fix them at the model level, early in the design cycle, where changes are easier and cheaper to make. More importantly, it holds the promise of improving the safety of the system through verification. Model-based techniques currently used in the medical device industry rely on semiformal approaches such as UML and Simulink [9] and thus do not allow developers to fully utilize the benefits of model-based design.

Our Approach. Here we will discuss a model-based development process, illustrated by two recent case studies: an infusion pump [42] and an implantable pacemaker [38]. The process, shown in Fig. 2, relies on well-known formal modeling and analysis tools, UPPAAL [10] and TIMES [6], to develop and verify the model of a system and generate code from it.

We begin with system requirements, which are given as natural language or using informal state machine notation. Requirements may also contain tolerance, with which timing constraints must be satisfied by an implementation. Based on the requirements, we develop a system model in UPPAAL. We also formalize requirements in a variant of

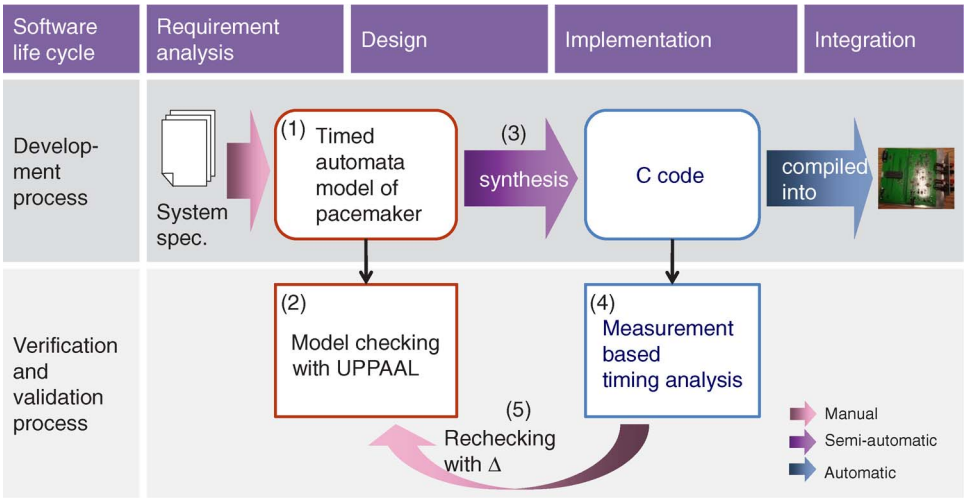


Fig. 2. ***Development process in the pacemaker case study.***

the CTL temporal logic supported by the UPPAAL model checker. Then, we perform verification of the UPPAAL model with respect to the formalized requirements. The obtained model is an ideal one, in that it satisfies the requirements under the assumption that computation performed by the system is instantaneous. From the verified model, we perform code generation using the TIMES tool and adapt the code to the selected execution platform. We then perform validation of the resulting system implementation with respect to the system requirements. In addition to testing functional behavior of the system, we also check that timing constraints are within tolerances specified in the requirements.

Validation results may show that tolerances are not observed due to nontrivial computation time in the implementation. In this case, we identify the source of the violation and modify the model according to measurements on the implementation, obtained during the validation experiments. We then perform another iteration of the process, reverifying the model, performing code generation again, and repeating the validation. It is possible that several iterations are needed until all constraints are met. However, in our case studies, we did not have to cycle more than once. When the process completes, we obtain a software implementation of the system that runs on the chosen platform and has been validated against the system requirements.

Generic PCA Infusion Pump: Patient-controlled analgesia (PCA) infusion pumps are widely used for pain control of postoperative patients. PCA pumps deliver opioid drugs, which put the patient at risk of respiratory depression and death in case of overdose. PCA pumps therefore are subject to stringent safety requirements that aim to prevent overdose. The generic PCA (GPCA) project, a joint effort between our group and researchers at the FDA, aims to develop a series of publicly available artifacts that can be used as guidance for manufacturers. In the first phase of the project, a collection of documents has been developed, including a hazard analysis report, a set of safety requirements [39], and a reference model of the state controller. The model concentrates on the state transitions in the controller software and abstracts away much of the functional computation performed by the pump (for example, estimation of the remaining drug volume to be infused).

Based on these documents, we studied a model-driven implementation of a PCA infusion pump controller software. We used the Hospira LifeCare PCA pump platform, as the platform of choice in this regard. The platform contains a microcontroller-equipped pump motor, several sensors for environmental conditions, and a buzzer for sounding alarms. Controller software is deployed on a OMAP3530 processor running Linux OS, which communicates with the motor microcontroller over a serial line. The user interface is implemented using an Android application on a smartphone.

Following the process described above, we formalized both the model and the requirements in UPPAAL, performed formal verification of the model, then generated code using TIMES. To validate the generated code, we performed conformance testing using a testbed that monitors the execution of the implementation and compares it with the corresponding model execution. No violations were observed during testing.

As part of the formalization process, we categorized safety requirements according to their precision and level of abstraction: 1) requirements that are detailed enough to be formalized and verified on the UPPAAL model; 2) requirements that are beyond the scope of the UPPAAL model; and 3) requirements that are too imprecise to be formalized. Only 20 out of 97 requirements fell into the first category. Most of the requirements in the second category concern the functional aspects of the system that

are abstracted away by the UPPAAL model. Implementation of these functional aspects, such as the remaining drug volume estimation, is performed outside the model-based process. Thus, these requirements can be validated on the implementation. An example from the third category is "flow discontinuity at low flows should be minimal," which does not specify what is a low flow or what discontinuity can be accepted as minimal. This example shows the importance of a model-based process not just for software design, but also for requirements engineering. Through formalization, we are forced to identify ambiguous requirements like the one shown above and provide feedback to the requirements engineers.

Pacemaker Challenge Problem: The second case study was motivated by the pacemaker challenge, the certification challenge problem issued by the software certification consortium (SCC) [61]. The challenge involves the development of pacemaker controller software that is formally verified for compliance with the timing requirements released by *Boston Scientific*.

A cardiac pacemaker is an electronic device implanted into the body to compensate for irregularities in the intrinsic heart rhythm by delivering electrical stimuli, called *paces*, to the heart. The pacemaker may also detect natural heart activity, called *sense* signals. Timing requirements for the pacemaker operation are given by a number of properties known as timing cycles. The applicable timing cycles depend on the operating mode of the pacemaker. The operating mode is set by a physician before the pacemaker is implanted, depending on the patient condition. The mode describes what sensing and pacing is performed and relationship between them. In this paper, we considered the mode, in which the pacemaker performs ventricular sensing and ventricular pacing with inhibition (VVI). That is, pacing will occur at regular intervals, unless heart activity in the ventricle is sensed. A sense will inhibit the next pace, allowing the heart to beat on its own.

The timing cycles applicable in the VVI mode make use of the lower rate limit interval (LRI) and the ventricular refractory period (VRP). The LRI specifies the maximum time interval that can elapse between any two cardiac events (senses or paces). During the VRP, which begins after every pace, sensing has to be turned off to avoid pace-induced false senses. In addition to prescribing the timing cycles, the pacemaker requirements specify tolerances with which the cycles must be adhered to.

We used the process described above to implement the pacemaker controller [38]. First, we created a model of the controller using timed automata in the UPPAAL tool, then we converted the applicable timing cycles into temporal logic properties and verified them using the UPPAAL model checker, followed by code generation. In this case, however, validation demonstrated that timing constraints were not satisfied within their tolerance levels. By analyzing the timing traces of the controller, we identified operations that contributed most to the property violation and measured their execution times. We modified the UPPAAL model by tightening transition guards, making the offending operations start earlier. After reverifying the model and generating the code again, the controller implementation passed the validation phase.

Future Work. The case studies demonstrated that, while the overall process achieves good results in practice, it is somewhat *ad hoc* when iterations of the process occur. Further work is needed to identify, which changes to the model would make the process converge faster, or to quickly detect that the timing requirements are not implementable and no amount of model modification will succeed. In general, more rigorous development processes are needed, providing stronger guarantees for timing and other nonfunctional properties.

IV. SOFTWARE PLATFORMS FOR MEDICAL DEVICE INTEROPERABILITY

Engineering MCPS goes beyond considering the individual medical device: MCPS will consist of networks of medical devices and computer systems cooperating to provide care to patients. Conceptually, the set of device types and the algorithm which defines how those devices should interact in a given clinical scenario is a virtual medical device (VMD). VMDs can be instantiated into a *VMD instance* by coupling specific[1] network-connected devices with the appropriate clinical algorithm executing on a computer. The software artifact that contains the list of required devices and the executable clinical algorithm can be thought of as a *VMD App*.

To manage the instantiation and shutdown of the VMD, there should be some systems present on the hospital network. For example, the medical device plug-and-play (MD PnP) interoperability initiative [26] has proposed an *interoperability platform* and architecture known as the integrated clinical environment (ICE) which would serve that purpose. Such a "plug-and-play" mechanism can be used to support the following clinical workflow.

1) The clinician decides on the treatment plan for the patient.
2) The clinician selects medical devices used to apply the treatment.
3) The clinician assembles the treatment system by connecting the devices to an ICE supervisor computer and to the patient.
4) The clinician starts the required VMD App, which will automatically bind the physical devices into the VMD instance and orchestrate interaction among the devices to execute the treatment plan.

[1]The specific devices must possess the capabilities required by the given VMD.

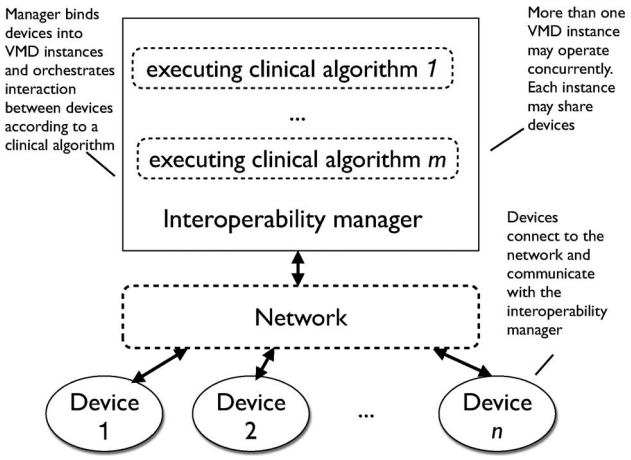


Fig. 3. ***Relationship between devices, interoperability manager, and clinical algorithms.***

At the moment steps 3 and 4 are infeasible, because out-of-the-box medical devices do not have the ability to communicate with each other. Further, they are neither interoperable (i.e., expose interfaces for remote control to the network), nor safely composable. This also severely limits the choices for the first and second steps, because the clinician must take the availability and quality of only single devices into consideration when deciding on the treatment plan.

Our Approach. As shown in Fig. 3, a critical component in interoperability platforms is the *interoperability manager*. The interoperability manager tracks which medical devices are connected to the network, binds connected medical devices into VMD instances, provides an execution environment for a VMD's workflow algorithms, detects device or network faults, and ensures that concurrently executing VMD instances do not interfere with one another. We have been building an open-source, prototype, interoperability manager known as the medical device coordination framework (MDCF) [44].

Medical Device Coordination Framework Overview: The MDCF is middleware designed to facilitate and manage the composition of medical devices and clinical algorithms into VMD instances. The middleware consists of a server process that runs on a computer and a lightweight communications library that can be incorporated into the software of the individual medical devices. The server process and communications library work in tandem to provide the following.

1) *Publish–Subscribe Messaging Service*: Data between the MDCF and the devices are published as messages to *topics*. Publish–subscribe communication facilitates easy sharing of data (e.g., physiologic data from one device can easily be shared among multiple concurrent VMD instances).
2) *Device Management*: The MDCF only allows devices approved by the MDCF administrator to associate (connect) with the middleware. The status of associated devices is tracked via a heartbeat mechanism.
3) *VMD App Management*: The MDCF facilitates the instantiation and deactivation of VMD instances. Clinicians instantiate a VMD by selecting a VMD App and the currently connected devices to use in the VMD. The MDCF will load the VMD App's clinical algorithm into a virtual machine and then bind the selected devices to the algorithm. The VMD deactivates a VMD instance by notifying each device that it is no longer bound and by releasing any resources used by the clinical algorithm.

Additionally, the MDCF provides an integrated development environment (IDE), which is a separate program VMD App developers can use to design and implement VMD Apps as well as define the logical interfaces (i.e., capabilities) of MDCF devices. In particular, developers can use the IDE to do the following.

1) *Define Device Types*: A device can be abstractly described in terms of the data it publishes (output ports) and the data it subscribes to (input ports).
2) *Automatically Generate Code*: The IDE can automatically generate the communications library for a specific device type. The generated code handles all the communications between the MDCF and the device (including an implementation of the MDCF association protocol). Because the generated code conforms to the MDCF specification for the device type, device developers only need to write the "adapter code" required to convert data between the MDCF data format and the device's native data format.
3) *Specify VMD Apps*: VMD App developers can define their application by specifying what device types are used in the VMD, what clinical algorithms are used, and what topics each device and algorithm should subscribe to. Fig. 4 shows how

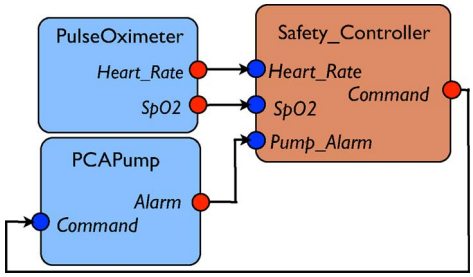


Fig. 4. ***Dataflow specification for a closed-loop PCA application.***

the closed-loop PCA application (described later in Section VI) looks like in the IDE's graphical VMD App editor.

While the MDCF is a prototype and under development, we have used the MDCF to implement a number of different medical device integration scenarios such as a closed-loop PCA application (see Section VI) using a combination of real and simulated devices [43], and a *smart alarm* (see Section V) for postcoronary artery bypass graft surgery patients [45].

Future Work. A number of open problems remain to be addressed in VMD instantiation. Support for real-time communication, which requires runtime allocation of processor and network resources to VMD Apps, is currently missing. A large part of the challenge is to implement such allocation in hospital communication network, which are open to various sources of interference from other traffic. We also need to establish a sufficient level of confidence in both the interoperability framework, such as MDCF, and VMD Apps. For this, verification of association and communication protocols used by MDCF is necessary, as well as establishing conformance of MDCF software implementation to formal design specifications. A process similar to the one discussed in Section III should be used for the development of high-confidence VMD Apps and their instantiation over MDCF. Additionally, open challenges in medical device interoperability include security and privacy concerns (see Section VIII) and certification of VMDs (see Section VII).

V. SMART ALARMS AND CLINICAL DECISION SUPPORT

Achieving medical device interoperability will allow medical information to be streamed from multiple devices into central locations in real time. This presents unique opportunities to significantly improve clinical care. Modern hospital rooms are commonly equipped with many such medical devices, currently used to continuously monitor their patients' vital signs. These vital sign monitors give clinicians a window into the patient's state and can be configured to alert clinicians to a deterioration in state. Most medical devices currently in use can only be configured with threshold alarms, i.e., alarms which activate when a vital sign crosses a predefined threshold. While threshold alarms can be vital in the timely detection of emergency states, they have been shown to be unscientific [47], cause a high number of alarms [33], and have a high rate of false alarms [17], [35], which fatigues caretakers [60] and leads them to ignore or turn off many alarms [21]. This has been shown to decrease quality of care [17], [20], [34]. Also, these alarms only alert clinicians to the fact that some threshold was crossed; they fail to provide any physiologic or diagnostic information about the current state of the patient that might help reveal the underlying cause of the patient's distress.

Creating MCPSs which stream real-time medical information from different devices and combine it with information from the patient's health record would improve the accuracy and usefulness of alarms [11], [13], [14]. Such systems could then be equipped to automatically suppress irrelevant alarms and provide summaries of the patient's state, as well as predict future trends in the patient. These MCPSs, realized as VMDs, would act as high accuracy *smart alarms*, which would alert clinicians to deterioration in the patient's state quickly and precisely, while providing them with access to the data that evidences the deterioration [34]. There have been many calls for a focus on evidence-based medicine as standard practice [22], [57], and smart alarms would help to satisfy these calls.

Smart alarm systems in particular require achieving some level of context-aware computational intelligence in MCPSs. Relevant information from multiple medical device data streams must be extracted and filtered [15], [34] and used in concert with a patient model to create a context-aware clinical "picture" of the patient. Developing context-aware computational intelligence is difficult. Possible solutions, such as encoding hospital guidelines, extracting mental models from medical professionals, and learning the models statistically from data all pose unique challenges.

Many efforts have been made to improve the accuracy of threshold alarms [16], [23], [52], [58]. Likewise, many clinical decision support systems, which are inherently evidence based, have been shown to hold promise in improving care [25], [32]. Initial successes in the area highlight the need for a cohesive, unified effort to improve all alarms used in hospitals.

Our Approach. Achieving context awareness in MCPSs requires the ability to preprocess and store patient data from patient streams. Techniques can be simple, such as downsampling or capturing trends, or they can be complex, such as using time series analysis to extract meaningful characteristics from a waveform. These techniques are well understood and can be employed to this end. Determining the best technique to use in any application, however, is difficult.

Data thus acquired must be compared with some sort of clinical model to be used in an intelligent fashion. As with preprocessing, this multitude of available techniques makes choosing the best technique difficult. Additionally, few machine learning techniques have been rigorously analyzed in the context of medicine. Chosen algorithms must be equipped to handle missing values and effectively account for the passage of time.

Generic Smart Alarm: We are addressing these challenges by building smart alarm systems subscribing to a generic architecture which is flexible enough to rapidly prototype reconfigurable and verifiable systems. This architecture consists of: several stages of preprocessing modules which process the raw data from the medical

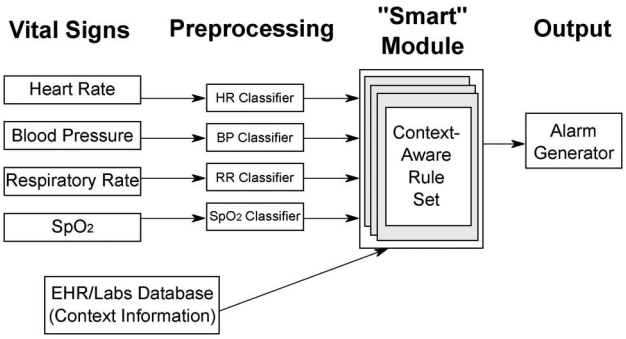


Fig. 5. ***Generic smart alarm architecture, instantiated as a smart alarm for CABG patients.***

devices and deliver it to the inference modules; a stage of inference modules which combine these data streams to produce some high-level output (e.g., an alarm level); and a visualization component. This architecture is shown in Fig. 5.

We are working closely with doctors and nurses at the Hospital of the University of Pennsylvania on specific implementations of the generic smart alarm. The smart alarm is designed to target areas of the ICU in which alarms are perceived as inadequate. We also plan to introduce smart alarms in areas where no alarm exists, owing to the difficulty to diagnose some particular alarmable patient state. Each of these implementations has begun to shed light on possible solutions to the challenges outlined above.

Smart Alarm for CABG Patients: After undergoing artery bypass graft (CABG) surgery, patients are at high risk of physiologic instability [49] which causes a high level of false alarms. To attempt to improve the accuracy of alarms in this domain, and to begin testing the generic smart alarm architecture, we created a straightforward instantiation of the GSA, using simple preprocessing and inference modules. We interviewed ICU nurses to create preprocessing modules which classified four major vital signs commonly monitored in the ICU (heart rate, blood pressure, blood oxygen saturation, and respiratory rate) into categories based on ranges ("low," "normal," "high," "very high," etc.). Afterwards, nurses determined rules that identified combinations of these vital sign categories which they deemed would be cause for concern. The smart alarm monitors a patient's four vitals, classifies those into categories, and searches the rule table for a corresponding alarm level to output. Combining vital signs produced a 57.13% reduction in false alarms generated without suppressing any true alarms [45].

Seizure Smart Alarm: Brain tissue oxygen monitors are currently utilized in a threshold-based fashion. Some practice guidelines suggest that a seizure is a potential culprit when brain tissue oxygen crosses a particular threshold. We have conducted preliminary studies which seem to indicate that this threshold may not be substantiated [54]. We are currently integrating multiple vital signs including brain tissue oxygen to create a "smart alarm" for seizures in the context of neurocritical care which goes beyond simple thresholds.

Vasospasm Smart Alarm: After aneurysmal subarachnoid hemorrhage, patients are kept in the ICU for up to 14 days to monitor for cerebral vasospasm (VSP), a narrowing of the blood vessels in the brain. The VSP condition, if undiagnosed or untreated, can lead to cerebral ischemia and neurologic dysfunction. While there are clinical factors which increase suspicion for VSP, the ability to define onset of this clinical syndrome is made difficult by the poor sensitivity of available tests. The only definitive measure of its presence is a cerebral angiogram, which is an invasive and resource-intensive study whose repetition is limited by radiation and contrast exposure to the patient. There is benefit to detecting VSP early, and risk to the patient from overtesting with cerebral angiogram. Integrating signals from multiple patient monitors, we strive to reduce the number of false alarms for VSP, as well as enable discovery of significant features that would improve the timeliness of diagnosis.

Future Work. As mentioned above, these sorts of smart alarms have been developed in the past, and most have been shown to improve care when utilized in hospitals. Few of these systems, however, have gained widespread use, due to several shortcomings.

1) *Data*: Current systems often combine only a few vital signs, limiting their scope. Most do not address issues with sparse data, or with capturing data not collected electronically. Significant challenges still exist in determining which patient model generation technique is most advantageous in any given context. Future work must expand the number of vital signs considered, and justify choice of model generator.
2) *Workflow*: Due to their experimental nature, published smart alarm systems are often highly complex and rarely incorporate user-centered design. Work is needed to ensure that smart alarm systems are clear and simple to use, which will help to justify their integration into clinician workflow.
3) *Practicality*: These systems are often domain and hospital specific, making reuse in wider contexts difficult. Additionally, these systems are rarely certified to be safe. Significant challenge still exists in understanding what safety means in the case of "smart" systems and in creating systems that are both safe and reusable.

Future smart alarms must take the form of a tool to be utilized to improve patient care, and not constitute a chore to be completed or a replacement for physician reasoning [59]. Smart alarms, however, have the potential to go beyond advisory roles.

VI. PHYSIOLOGICAL CLOSED-LOOP SYSTEMS

In smart alarm systems, physiological data are integrated and processed to provide clinical decision support. Integrated physiological data can also be used to directly control therapeutic delivery devices, forming a physiological closed-loop system. Automatic controllers have been successfully deployed in many safety critical systems, e.g., auto pilot in avionics and active cruise control in automobiles. In patient care, it is possible to construct a physiological closed-loop system by continuously monitoring patients' states, automatically reconfiguring delivery devices, and only alerting caregivers if patients' states divert from the normal range. Caregivers can then concentrate on making important clinical decisions, reducing the chances of missing critical events, thereby improving patient safety.

Closed-loop control has been deployed in some medical applications, but mostly in implantable devices such as cardioverter defibrillators, and other special purpose devices that do not need to be interconnected with other devices for routine operations. This need not be the case. A physiological closed-loop system can be modeled as a VMD and can be built in a cost-effective way by networking existing medical devices, such as infusion pumps and vital sign monitors. However, such a system also introduces new hazards that need to be systematically identified and mitigated.

One critical issue in applying the model-driven development to such systems is the patient modeling. There has been much work in patient modeling; for example, glucose–insulin kinetics have been extensively studied and modeled. Several control strategies, such as the model predictive control [46], [54], have been developed and evaluated on these specific biochemical models. However, most previous studies [12], [18] on physiological control assume no communication failures or delays. Our work, on the other hand, considers failures that networked closed-loop systems may suffer from in practice.

Fig. 6 shows how medical devices can be interconnected to form a physiological closed-loop system. We first give a high-level overview of two clinical cases that can benefit from closed-loop systems, and we will revisit them to address the technical challenges after introducing our general approach. The systems in the two cases share the same structure shown in the figure.

Closed-Loop Patient Controlled Analgesia (PCA): As mentioned in Section III, a major safety concern in PCA pump use is that an overdose of analgesic can cause respiratory failure. However, the existing safety mechanisms are considered insufficient to cover all possible scenarios [51]. A closed-loop solution to address the safety issues of PCA pump use is proposed in our previous work [7]. Here, a pulse oximeter is used to continuously monitor two respiratory-related vital signs, heart rate

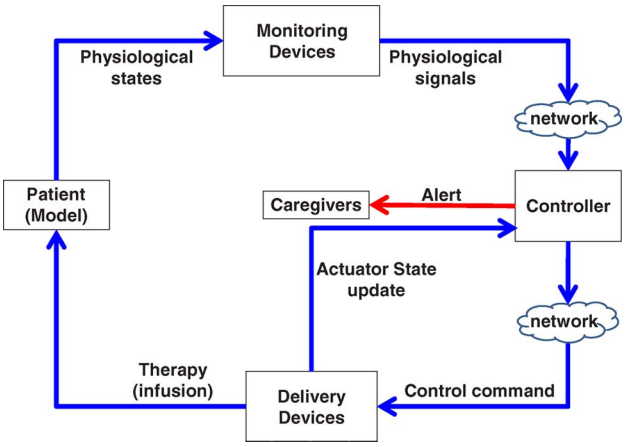


Fig. 6. ***PCA/BG closed-loop system.***

(HR), and blood oxygen saturation (SpO_2), and transmit the readings to a controller. The controller can stop the PCA infusion if it detects possible respiratory failures based on the HR/SpO_2 readings, and thus overdosing is prevented.

Closed-Loop Blood Glucose (BG) Regulation: Diabetics and some ICU patients depend on external insulin and glucose administration to maintain their BG level within a reference range, e.g., 70–130 mg/dl [1]. Traditionally, nurses take glucose measurements regularly and manually adjust the insulin infusion rate or administrate glucose. The problem is that the interval between two check points may be rather long, which limits the quality of control leading to severe oscillations of BG levels. In addition, the manual measurement and adjustment process is prone to human errors. In a closed-loop BG control system, a controller continuously monitors the BG and adjusts the insulin infusion rate. Caregivers are alerted if an adverse event such as hypoglycemia occurs. Such a closed-loop system can improve the quality of BG control.

In the above two cases, networked closed-loop systems can be used to improve clinical care, but networks also introduce new hazards that could compromise patient safety. Our research goal is to develop networked physiological closed-loop systems that assure patient safety. The key issue here is to identify and mitigate hazards introduced by the networked closed-loop systems, and ensure that the hazardous situations do not happen.

Our Approach. For the design of a safe physiological closed-loop system, we follow the iterative verification and validation approach shown in Fig. 7. First, we identify concrete use cases. We then model individual components of the system as shown in Fig. 6: the patient, monitoring and delivery devices, and network. The patient model is developed by implementing physiological models on verification and simulation tools, such as UPPAAL and Simulink. We model the input–output behavior of

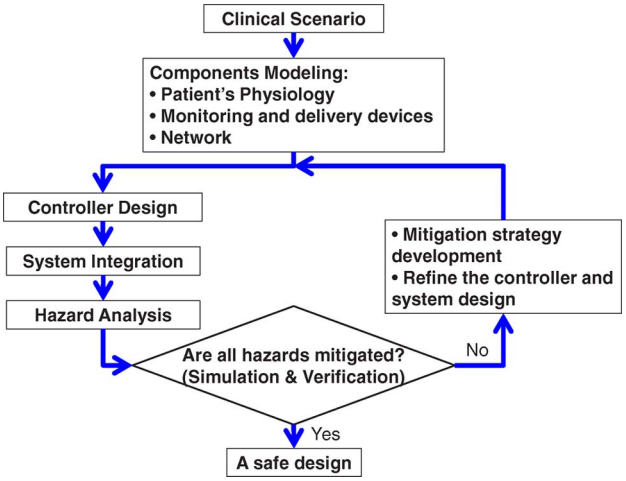


Fig. 7. ***Development of a distributed physiological closed-loop system.***

monitoring and delivery devices by considering the measurement and delivery errors that realistic devices may exhibit [41]. For example, to model a glucometer that has 10% measurement errors, our glucometer model introduces 10% errors onto the BG value it gets from the patient model, and transmits the resulting value to the controller. The network model describes the statistical behavior of network communication.

Next, we model a controller and design the entire system by connecting and defining interfaces between individual components. We then perform a hazard analysis and identify possible causes for each hazard. For example, in the PCA case, a possible cause of overdosing is that a "stop" command fails to reach the PCA pump due to network failure. We systematically identify such failure conditions and check whether all safety properties are satisfied in all scenarios, by running simulation on Simulink and formal verification on UPPAAL. We revise controller and system designs until all hazards are proved to be properly mitigated. The final step is instantiating our safe design of physiological closed-loop VMD as a VMD instance.

We next describe how we have applied this design process to the PCA and BG case studies.

Closed-Loop PCA: We have developed a patient model from [48] to represent the effect of drug level to HR and SpO_2. The model can be tuned to capture the variation of patient dynamics. We have constructed a Simulink model to precisely capture the continuous dynamics of the patient model and validated it by simulation. To formally verify the timing properties of the model, we also constructed a UPPAAL model that abstracts continuous dynamics, but preserves the timing behavior of components, which allows us to reason about the maximum end-to-end delay in delivering commands to the pump. We identified two types of failure in the PCA closed-loop system:

- sensor failures, e.g., the pulse oximeter leads get detached from the patient so that the controller cannot receive correct HR/SpO_2 readings;
- network failures, in which case the controller cannot receive any readings from the sensor and the pump cannot receive any commands from the controller.

To mitigate hazards due to these failures, we refined the system design so that open-loop stability is guaranteed; that is, if the controller cannot receive readings or the pump cannot receive control commands, the system can automatically go into a safe mode to prevent overdosing. One possible solution to achieve such open-loop stability, or fail-safe operation, is to let the controller instruct the PCA pump the duration of each drug delivery requested by the patient. The controller calculates this duration at run time based on the difference between the patient's current drug level and the safety limit. Therefore, it is guaranteed that the patient cannot be overdosed within the instructed time duration. Doing so, even if the sensor or network fails, the pump will stop infusion based on the last duration command it received, thus the system can fail-safe. Another possible solution is let the controller set a maximum drug dosage.

Closed-Loop BG Control: We have implemented the glucose–insulin patient model introduced in [55] in Simulink. As a start point of controller design, we have modeled and implemented several BG control guidelines being used at the Hospital of the University of Pennsylvania. We have run closed-loop simulations by connecting the patient model with the guideline-based controllers. Our preliminary simulation results show that long intervals between two check points may indeed limit the quality of control and result in severe overshooting of BG trajectories.

We have found that the complexity of hazard analysis is closely related to control objectives. Specifically, the objective of maintaining BG within a certain range makes the hazards identification and mitigation in the BG case more complicated than the PCA case. This is because there is no trivial fail-safe mode for BG control (e.g., simply stopping infusion is no longer a safe default option), and a mitigation strategy design is more challenging because both hyperglycemia and hypoglycemia need to be avoided.

Future Work. Validation of physiological closed-loop systems remains a major challenge. Ultimately, clinical outcomes delivered by a closed-loop system need to be compared with current caregiver-centric approaches. We are exploring the use of SimMan [2], which provides a controlled, realistic caregiving environment with patient simulator. More generally, we will explore the notion of *virtual clinical trial*, which so far has been considered only in drug development. Also, the role of a caregiver in a closed-loop system needs to be studied more carefully.

After all, we are not aiming at fully autonomous systems that exclude humans. We expect that work on smart alarm systems (see Section V) will provide useful insights.

VII. CERTIFICATION AND REGULATORY ISSUES

Like most safety-critical system, MCPSs are subject to regulatory oversight through a certification or approval process. The fundamental goal of certification of MCPSs is to assess safety and effectiveness of MCPSs. The traditional process-based regulatory regime used currently by the FDA to approve medical devices is becoming inadequate for the MCPS complexity [31]. A new, evidence-based regulatory regime is being put in place. Within the infusion pump improvement initiative [63], the FDA now requires assurance cases as part of the documentation submitted for approval. We can expect that similar requirements will be extended to MCPSs in general.

An important part of the challenge is software certification and ways to incorporate it into the regulatory approval process for the device as a whole. Medical devices have increasingly large amounts of software, performing various monitoring and care delivery tasks. Software-specific risks are not as well understood, and evidence of software quality is harder to evaluate. Current design practice places verification and certification at the end of the design cycle, when it is frequently too late to change design choices. As medical devices become more complex and more interconnected, it is becoming increasingly evident that verification and certification should be incorporated in early design stages. This can be done in two ways: on the one hand, the "design for verification" approach [5] can help verification techniques scale better and make generation of verification evidence easier; on the other hand, model-based generative techniques can be used to enable one to perform verification early in the design and then extend the guarantees provided by verification to the implementation through code generation.

Throughout the domain of software-intensive embedded and CPS systems, a new regulatory approach to certification has been advocated [36], based on collecting and reviewing evidence that the system achieves its goals. Model-driven techniques can help with the transition to evidence-based certification, from the current process-based approach. Using compositional modeling techniques and assume-guarantee reasoning may enable *incremental certification*, which would allow us to recertify MCPSs after component upgrades without reconsidering the whole assurance case from scratch.

Our Approach. Model-based development processes produce a number of artifacts that can be used as evidence of system quality. The artifacts include models, formalized properties, results of verification and testing, etc. Assurance cases have been suggested for organizing the generated evidence for the purpose of regulatory approval or certification [36]. An assurance case is a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims about a system's properties are adequately justified for a given application in a given environment [4]. Assurance cases hold the promise of both reducing certification costs and improving the quality of certification by tying it to evidence. Yet, there are few commonly accepted ways of constructing assurance cases. A poorly structured assurance case, however, can hamper the evaluation process, rather than help it [66]. Clearly, there is no "one-size-fits-all" structure, and software developed through different processes is likely to require different arguments about its safety. In [37], we aimed to discover appropriate assurance case structures for model-driven development.

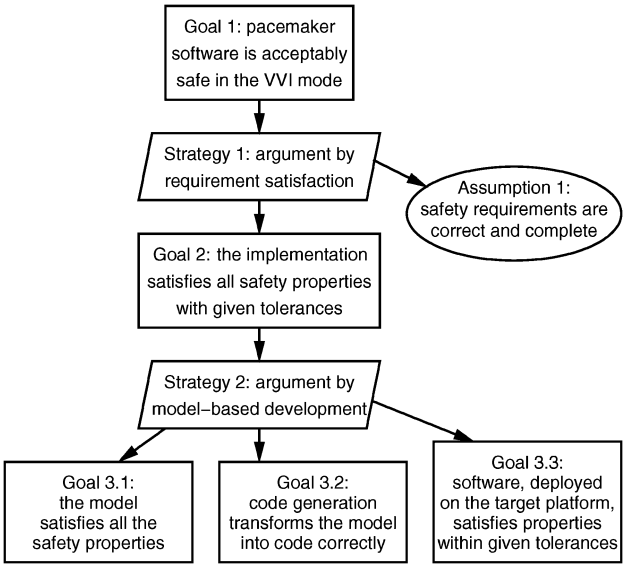


Fig. 8. ***Top-level claims of the pacemaker assurance case.***

Using the pacemaker case study described in Section III, we constructed an assurance case for the controller developed by our iterative-model-based process. The structure of the assurance case, shown in Fig. 8, reflects the main steps of the process. The top-level claim of the assurance case is that the pacemaker software is acceptably safe to operate. The two main subclaims decompose the argument into assessment of the system requirements and assessment of the requirements satisfaction. That is, we argue that requirements guarantee safety, and once that is established, argue that the developed system satisfies the requirements. Our case study, however, did not include development of the requirements, and thus we introduce an assumption that the requirements are adequate and concentrate on the second subclaim. It states that the code satisfies the timing properties with the specified tolerance. We then decompose the claim further into three subclaims according to the steps of the process: 1) the model satisfies

the timing properties, demonstrated by the UPPAAL verification results; 2) code generation preserves model properties, demonstrated by the correctness proof of the TIMES tool algorithm; and 3) the generated code exhibits the required timing properties with the prescribed tolerance, demonstrated by measurements during testing.

Future Work. Assurance Case Templates: The constructed assurance case captures the source of our confidence in the design as well as in the model-based process we used. Ultimately, we would like to develop assurance case templates for various development processes in order to simplify construction of cases for future systems developed through the same process. An important aspect of the assurance case framework, which remains a significant challenge for their application in practice, is evaluation of assurance cases. We need to develop means of rigorous, if informal, ways of quantifying the level of confidence delivered by an assurance case. This will be an important direction of our future work.

Certification of Virtual Medical Devices: Certification of virtual medical devices is another major challenge of MCPSs. It requires a fundamental change in the regulatory process. Standalone medical devices are certified (or approved) by a rigorous analysis of the device's design, manufacturing process, and final testing. However, VMD instances are instantiated in hospitals in different ways and cannot be certified individually. Ideally, a VMD should be certified as a specification, so that as long as a VMD instance is instantiated according to the respective VMD, it is guaranteed to be safe. Note also that VMD instances are not built from scratch, but from standalone medical devices, which were themselves certified, and an assurance case for a VMD instance should be able to rely on these certificates. A vision for VMD instance certification has been put forth under the names of compositional, runtime, or just-in-time certification [56]. However, to this date, the vision remains largely unrealized and requires further research. A regulatory framework based on third-party certification approach for VMDs is proposed in [29]. It lays out primary verification tasks associated with each VMD component and tool support necessary for the verification and certification activities. The framework is supported by a security and authentication layer in the interoperability platform that establishes trust in the safety and correctness of VMDs used in building the VMD instance.

VIII. OPEN ISSUES

In this section, we present some of the open issues that still require addressing in the domain of MCPS.

Security and Privacy. While interoperability capabilities allow medical devices to be incorporated into MCPS, acquiring functionality that was never possible previously, they also open the door to a host of security and privacy concerns [3]. An attacker who penetrates an MCPS network has the potential to harm or kill patients by reprogramming devices [27]. The increased autonomy of MCPS with closed-loop control, automated therapy delivery, and alarm capabilities has the potential to exacerbate the problem. In general, when attacking an MCPS, adversaries can choose from four classes of targets [8].

- *Patient*: An attacker directly targets the patient's health. This is usually achieved by targeting the sensing, processing, communication, and treatment delivery aspects of the MCPS. One example might involve an attacker programming an infusion pump to administer a larger than necessary dose of medicine.
- *Data*: An attacker accesses an individual patient's health data from the MCPS in an unauthorized manner. The consequence is loss of patient privacy leading to potential discrimination and abuse [64].
- *Device*: An attacker mounts a denial of service (DoS) on the MCPS in some form so that it cannot perform its task, thus limiting device availability. This can also result in loss of privacy in systems that are designed to fail-open as suggested in [19].
- *Institution*: The goal is to compromise the interaction between the MCPS and the internal network of the institution it is deployed in to obtain access, at a large scale, to patient data or network operational information.

The approach usually taken by most device manufacturers today is to limit the functionality that can be invoked through the network interface. In most cases, the device can send out data, such as sensor readings or event logs, but not accept commands from the network. Although such an approach improves security of the system, it severely limits the ability to deploy closed-loop scenarios. In other cases, device manufacturers introduce proprietary security solutions and rely on "security through obscurity." This attitude has been shown to be problematic, as adversaries will always be able to break into such a system [24].

Recent years have seen the issue of medical device security addressed for different classes of medical devices such as implantable [19], [27] or interoperable devices [64]. However, in most of these cases, the focus is on specific aspects of MCPS operation, namely secure communication, and effective access control. Fundamentally, the challenge of targeting security for MCPS involves developing *flexible* and *open* solutions while addressing the following four issues: 1) minimizing the *overhead* that security solutions inevitably bring; 2) dealing with the *heterogeneity* of MCPS that precludes system-wide solutions; 3) improving *usability* (even transparency) of security solutions developed; and 4) considering *safety* implications of security solutions and decisions.

As a first step in securing MCPS, we are extending the MDCF to support encrypted communications between the devices and MDCF. Additionally, the middleware server needs to establish trust in the devices, and the devices need

to establish trust in the middleware (i.e., only known, certified devices should be bound into a VMD) [29]. Furthermore, the MDCF needs to ensure that the set of medical devices are indeed connected to the same patient. Eventually, more holistic solutions for MCPS that look at security in the larger context of their deployment in clinical workflows will need to be created.

Patient Modeling and Simulation. A closely related challenge is that of patient modeling. Patient models are needed for the design of closed-loop control, as well as for the safety analysis of scenarios. For example, the closed-loop PCA scenario requires a model of drug absorption by the patient body, as well as the relationship between the drug dose and concentration and patient vital signs, such as heart rates and respiratory rates. Pharmacokinetic models of drug absorption are known from the literature (e.g., [48]), and there is statistical data on the effect of the drug on vital signs. However, comprehensive models are too complex to be used in the design and analysis. Thus, development of new abstraction techniques is paramount for addressing this challenge.

User-Centered Design. Caregiver errors in using medical devices are a major source of adverse events [30], [65]. Undoubtedly, some of these errors are due to stress and overload that caregivers experience daily. Poor user-interface design also has been attributed for many of these errors. If a device is hard to operate, has a counterintuitive interface, or responds to user inputs in an unexpected manner, user errors are much easier to occur. Design and validation of medical devices needs to take into account user expectations. To use model-based design for interactive medical devices, we have to incorporate models of caregiver behavior. Such user modeling is a notoriously challenging problem. However, incorporating information about likelihood of certain actions into caregiver models opens the way for quantitatively reasoning about device safety.

Compositionality. Interoperable network-enabled medical devices will increasingly be composed into MCPS dynamically. Compositional reasoning is the only rigorous way to ensure safety of such systems. A particularly challenging problem is predicting the possibilities of unexpected interactions between devices in the system. For example, devices providing different treatments to the same patient may incur radio interference because of close proximity to each other. More importantly, treatments themselves can interfere with each other by affecting physiological responses [28]. MCPS designers should be aware of these interferences and ensure that the system providing a treatment is made aware of potentially interfering treatments through sufficient context information.

Continuous Monitoring and Care. One of the most important needs of modern medicine is to develop medical devices capable of providing continuous care (i.e., monitoring, decision support, and delivery of therapy). Such devices are expected to decrease healthcare cost by enabling alternatives such as home-based or ambulatory care. Caregivers can have a detailed picture of the patient's health at all times, enabling them to better tune the treatment provided. Such a system also allows for real-time notification in the event of emergencies and providing first-responders with accurate and complete information about the patient's health. Continuous care systems are being designed to monitor a plethora of ailments such as cardiovascular diseases [40], neurological problems [62], collecting meta-physiological state information (sleep, awake, fatigue) [55], circadian activity monitoring [67], and extreme environment medical monitoring (e.g., space) [50].

IX. CONCLUSION

In this paper, we considered MCPSs as a distinct CPS domain. Due to increasing societal pressures and new technological capabilities, the field of MCPSs is on the verge of a substantial transformation that will change the ways these systems are developed and approved, as well as expand features and strengthen safety guarantees the MCPS offer to caregivers and patients. This transformation will lead to a further increase in the complexity of MCPS and the degree of integration within them.

Thus, the domain of MCPS offers a unique set of challenges, distinct from any other CPS domain [31]. The challenges facing MCPSs are formidable, yet they present vast opportunities for research with immediate practical impact. We identified major challenges in MCPS development and discussed promising research directions that may help to overcome some of these challenges. We envision that these challenges will provide major opportunities for R&D communities in the next ten years. ■

REFERENCES

[1] American Diabetes Association, accessed Jun. 15, 2011. [Online]. Available: http://www.diabetes.org/living-with-diabetes/treatment-and-care/blood-glucose-control/tight-diabetes-control.html

[2] Laerdal SimMan, Jun. 17, 2011. [Online]. Available: http://www.laerdal.com/us/doc/86/SimMan

[3] M. J. Ackerman, L. P. Burgess, R. Filart, I. Lee, and R. K. Poropatich, "Developing next generation telehealth tools and technologies: Patients, systems, and data perspectives," *Telemedicine and e-Health*, vol. 16, no. 1, pp. 93–95, Jan./Feb. 2010.

[4] *ASCAD—The Adelard Safety Case Development (ASCAD) Manual*, Adelard, 1998.

[5] K. Alexander and P. Clarkson, "Good design practice for medical devices and equipment—Part II: Design for verification," *J. Med. Eng. Technol.*, vol. 24, no. 2, pp. 53–62, 2000.

[6] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "TIMES: A tool for schedulability analysis and code generation of real-time systems," in *Proc. 1st Int. Workshop Formal Modeling Anal. Timed Syst. (FORMATS 2003)*, 2003, pp. 60–72.

[7] D. Arney, M. Pajic, J. Goldman, I. Lee, R. Mangharam, and O. Sokolsky, "Toward patient safety in closed-loop medical device systems," in *Proc. 1st Int. Conf. Cyber-Physical Syst.*, Apr. 2010, DOI: 10.1145/1795194.1795214.

[8] D. Arney, K. Venkatasubramanian, O. Sokolsky, and I. Lee, "Biomedical devices and systems security," in *Proc. 33rd Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2011.

[9] U. Becker, "Model-based development of medical devices," *Proc. Workshop Comput. Safety, Reliability, Security (SAFECERT '09)*, vol. 5775. Berlin, Germany: Springer-Verlag, 2009, pp. 4–17, ser. Lecture Notes in Computer Science.

[10] G. Behrmann, A. David, and K. Larsen, "A tutorial on UPPAAL," *Formal Methods for the Design of Real-Time Systems (Revised Lectures)*, vol. 3185. Berlin, Germany: Springer-Verlag, 2004, pp. 200–237, ser. Lecture Notes in Computer Science.

[11] M. Borowski, M. Görges, R. Fried, O. Such, C. Wrede, and M. Imhoff, "Medical device alarms," *Biomed. Tech.*, vol. 56, pp. 73–83, 2011.

[12] B. P. Kovatchev, M. Breton, C. D. Man, and C. Cobelli, "In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes," *Diabetes Sci. Technol.*, vol. 3, no. 1, pp. 44–55, 2009.

[13] J. H. Burton, J. D. Harrah, C. A. Germann, and D. C. Dillon, "Does end-tidal carbon dioxide monitoring detect respiratory events prior to current sedation monitoring practices?" *Acad. Emergency Med.*, vol. 13, pp. 500–504, May 2006.

[14] M.-C. Chambrin, "Alarms in the intensive care unit: How can the number of false alarms be reduced?" *Critical Care*, vol. 5, pp. 184–188, 2001.

[15] S. Charbonnier and S. Gentil, "A trend-based alarm system to improve patient monitoring in intensive care units," *Control Eng. Practice*, vol. 15, no. 9, pp. 1039–1050, Sep. 2007.

[16] G. Clifford, W. Long, G. Moody, and P. Szolovits, "Robust parameter extraction for decision support using multimodal intensive care data," *Philosoph. Trans. Roy. Soc. A, Math. Phys. Eng. Sci.*, vol. 367, pp. 411–429, 2009.

[17] Clinical Alarms Task Force, "Impact of clinical alarms on patient safety," *J. Clin. Eng.*, vol. 32, no. 1, pp. 22–33, 2007.

[18] D. Bruttomesso, A. Farret, S. Costa, M. C. Marescotti, M. Vettore, A. Avogaro, A. Tiengo, C. Man, J. Place, A. Facchinetti, S. Guerra, L. Magni, G. Nicolao, C. Cobelli, E. Renard, and A. Maran, "Closed-loop artificial pancreas using subcutaneous glucose sensing and insulin delivery and a model predictive control algorithm: Preliminary studies in Padova and Montpellier," *Diabetes Sci. Technol.*, vol. 3, no. 5, pp. 1014–1021, 2009.

[19] T. Denning, K. Fu, and T. Kohno, "Absence makes the heart grow fonder: New directions for implantable medical device security," in *Proc. 3rd Conf. Hot Topics Security*, 2008, pp. 5:1–5:7.

[20] Y. Donchin and F. J. Seagull, "The hostile environment of the intensive care unit," *Current Opinion Critical Care*, vol. 8, pp. 316–320, 2002.

[21] J. Edworthy and E. Hellier, "Alarms and human behaviour: Implications for medical alarms," *British J. Anaesthesia*, vol. 97, pp. 12–17, 2006.

[22] Evidence-Based Medicine Working Group, "Evidence-based medicine. A new approach to teaching the practice of medicine," *J. Amer. Med. Assoc.*, vol. 268, pp. 2420–2425, 1992.

[23] J. M. Feldman and M. H. Ebrahim, "Robust sensor fusion improves heart rate estimation: Clinical evaluation," *J. Clin. Monitor. Comput.*, vol. 13, pp. 379–384, 1997.

[24] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications*. New York: Wiley, 2010.

[25] A. X. Garg, N. K. J. Adhikari, H. McDonald, M. P. Rosas-Arellano, P. J. Devereaux, J. Beyene, J. Sam, and R. B. Haynes, "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: A systematic review," *J. Amer. Med. Assoc.*, vol. 293, pp. 1223–1238, 2005.

[26] J. Goldman, R. Schrenker, J. Jackson, and S. Whitehead, "Plug-and-play in the operating room of the future," *Biomed. Instrum. Technol.*, vol. 39, no. 3, pp. 194–199, 2005.

[27] D. Halperin, T. Heydt-Benjamin, K. Fu, T. Kohno, and W. Maisel, "Security and privacy for implantable medical devices," *Perv. Comput.*, vol. 7, no. 1, pp. 30–39, Jan.–Mar. 2008.

[28] M. B. Happ, "Treatment interference in acutely and critically ill adults," *Amer. J. Critical Care*, vol. 7, no. 3, pp. 224–235, May 1998.

[29] J. Hatcliff, E. Vasserman, S. Weininger, and J. Goldman, "An overview of regulatory and trust issues for the integrated clinical environment," in *Proc. 3rd Joint Workshop High Confidence Med. Devices, Software, Syst. Med. Device Plug-and-Play Interoperability (HCMDSS/MDPnP 2011)*, 2011, pp. 1–10.

[30] R. W. Hicks, V. Sikirica, W. Nelson, J. R. Schein, and D. D. Cousins, "Medication errors involving patient-controlled analgesia," *Amer. J. Health-Syst. Pharmacy*, vol. 65, no. 5, pp. 429–440, Mar. 2008.

[31] High Confidence Software and Systems Coordinating Group, "High-confidence medical devices: Cyber-physical systems for 21st century health care. A research and development needs report," NCO/NITRD, Feb. 2009.

[32] D. L. Hunt, R. B. Haynes, S. E. Hanna, and K. Smith, "Effects of computerized clinical decision support systems on physician performance and patient outcomes: A systematic review," *J. Amer. Med. Assoc.*, vol. 280, pp. 1339–1346, 1998.

[33] M. Imhoff and R. Fried, "The crying wolf: Still crying?" *Anesthesia Analgesia*, vol. 108, no. 5, pp. 1382–1383, 2009.

[34] M. Imhoff and S. Kuhls, "Alarm algorithms in critical care monitoring," *Anesthesia Analgesia*, vol. 102, no. 5, pp. 1525–1536, 2006.

[35] M. Imhoff, S. Kuhls, U. Gather, and R. Fried, "Smart alarms from medical devices in the OR and ICU," *Best Practice Res. Clin. Anaesthesiol.*, vol. 23, no. 1, pp. 39–50, 2009.

[36] D. Jackson, M. Thomas, and L. I. Millett, Eds., "Software for dependable systems: Sufficient evidence?" Committee on Certifiably Dependable Software Systems, National Research Council, National Academies Press, New York, May 2007.

[37] E. Jee, I. Lee, and O. Sokolsky, "Assurance cases in model-driven development of the pacemaker software," *Proc. Int. Symp. Leveraging Appl. Formal Methods, Verification, Validation (ISoLA 2010)*, vol. 6416. Berlin, Germany: Springer-Verlag, Oct. 2010, pp. 343–356, ser. Lecture Notes in Computer Science.

[38] E. Jee, S. Wang, J. K. Kim, J. Lee, O. Sokolsky, and I. Lee, "A safety-assured development approach for real-time software," in *Proc. 16th IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl.*, Aug. 2010, pp. 133–142.

[39] R. P. Jetley and P. L. Jones, "Safety requirements based analysis of infusion pump software," *Proc. Workshop Softw. Syst. Med. Devices Services, held in conjunction with IEEE Real Time Syst. Symp.*, Tucson, AZ, pp. 21–24, Dec. 2007.

[40] Z. Jin, J. Oresko, S. Huang, and A. C. Cheng, "HeartToGo: A personalized medicie technology for cardiovascular disease prevention and detection," in *Proc. IEEE/NIH Life Sci. Syst. Appl. Workshop*, 2009, pp. 80–83.

[41] K. E. Anger and P. M. Szumita, "Barriers to glucose control in the intensive care unit," *Pharmacotherapy*, vol. 26, no. 2, pp. 214–228, 2006.

[42] B. Kim, A. Ayoub, O. Sokolsky, and I. Lee, "The safety-assured development of the GPCA infusion pump," *Proc. Int. Conf. Embedded Softw. (EMSOFT 2011)*, Taipei, Taiwan, Oct. 2011, (To Appear).

[43] 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 *Proc. ICSE Workshop Softw. Eng. Health Care*, 2010, DOI: 10.1145/1809085.1809086.

[44] A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. P. Jetley, P. L. Jones, and S. Weininger, "An open test bed for medical device integration and coordination," in *Proc. Int. Conf. Softw. Eng. Companion Volume*, May 2009, pp. 141–151.

[45] A. L. King, A. Roederer, D. Arney, S. Chen, M. Fortino-Mullen, A. Giannareas, W. Hanson, III, V. Kern, N. Stevens, J. Tannen, A. V. Trevino, S. Park, O. Sokolsky, and I. Lee, "GSA: A framework for rapid prototyping of smart alarm systems," in *Proc. 1st ACM Int. Health Inf. Symp.*, 2010, pp. 487–491.

[46] L. Magni, D. M. Raimondo, L. Bossi, C. D. Man, G. Nicolao, B. Kovatchev, and C. Cobelli, "Model predictive control of type 1 diabetes: An in silico trial," *Diabetes Sci. Technol.*, vol. 1, no. 6, pp. 804–812, 2007.

[47] L. A. Lynn and J. P. Curry, "Patterns of unexpected in-hospital deaths: A root cause analysis," *Patient Safety in Surgery*, p. 5, 2011.

[48] J. X. Mazoit, K. Butscher, and K. Samii, "Morphine in postoperative patients: Pharmacokinetics and pharmacodynamics of metabolites," *Anesthesia Analgesia*, vol. 105, no. 1, pp. 70–78, 2007.

[49] M. Mullen-Fortino and N. O'Brien, "Caring for a patient after coronary artery bypass graft surgery," *Nursing*, vol. 38, no. 3, pp. 46–52, Mar. 2008.

[50] C. Mundt, K. N. Montgomery, U. E. Udoh, V. N. Barker, G. C. Thonier, A. M. Tellier, R. D. Ricks, R. B. Darling, Y. D. Cagle, N. A. Cabrol, S. J. Ruoss, J. L. Swain, J. Hines, and G. T. A. Kovacs, "A multiparameter wearable physiological monitoring system for space and terrestrial applications," *IEEE Trans. Inf. Technol. Biomed.*, vol. 9, no. 3, pp. 382–391, Sep. 2005.

[51] T. K. Nuckols, A. G. Bower, S. M. Paddock, L. H. Hilborne, P. Wallace, J. M. Rothschild, A. Griffin, R. J. Fairbanks, B. Carlson, R. J. Panzer, and R. H. Brook, "Programmable infusion pumps in ICUs: An analysis of corresponding adverse drug events," *J. Gen. Internal Med.*, vol. 23, no. Supplement 1, pp. 41–45, Jan. 2008.

[52] C. Oberli, C. Saez, A. Cipriano, G. Lema, and C. Sacco, "An expert system for monitor alarm integration," *J. Clin. Monitor. Comput.*, vol. 15, pp. 29–35, 1999.

[53] S. Park, A. Roederer, R. Mani, S. Schmitt, P. D. LeRoux, L. H. Ungar, I. Lee, and S. E. Kasner, "Limitations of threshold-based

brain oxygen monitoring for seizure detection," *Neurocritical Care*, Apr. 2011.

[54] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, and M. E. Wilinska, "Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes," *Physiol. Meas.*, vol. 25, no. 4, pp. 905–920, Aug. 2004.

[55] M. D. Rienzo, F. Rizzo, G. Parati, G. Brambilla, M. Ferratini, and P. Castiglioni, "Magic system: A new textile-based wearable device for biological signal monitoring applicability in daily life and clinical setting," in *Proc. 27th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, 2005, pp. 7167–7169.

[56] J. Rushby, "Runtime certification," in *Runtime Verification*, vol. 5289, M. Leucker, Ed. Berlin, Germany: Springer-Verlag, 2008, pp. 21–35.

[57] D. L. Sackett and W. M. C. Rsenberg, "The need for evidence-based medicine," *J. Roy. Soc. Med.*, vol. 88, pp. 620–624, 1995.

[58] R. Schoenberg, D. Z. Sands, and C. Safran, "Making ICU alarms meaningful: A comparison of traditional vs. trend-based algorithms," in *Proc. Amer. Med. Inf. Assoc. Symp.*, 1999, pp. 379–383.

[59] E. H. Shortliffe, B. G. Buchanan, and E. A. Feigenbaum, "Knowledge engineering for medical decision making: A review of computer-based clinical decision aids," *Proc. IEEE*, vol. 67, no. 9, pp. 1207–1224, Sep. 1979.

[60] S. Siebig, L. Juhls, M. Imhoff, U. Gather, U. Scholmerich, and C. Wrede, "Plug-and-play for medical devices: Experiences from a case study," *Crit. Care Med.*, vol. 38, no. 2, pp. 451–455, 2010.

[61] *Pacemaker Formal Methods Challenge*, Software Quality Research Laboratory, McMaster University, Hamilton, ON, Canada, accessed Jun. 6, 2011. [Online]. Available: http://sqrl.mcmaster.ca/pacemaker.htm

[62] M. Sung, C. Marci, and A. Pentland, "Wearable feedback systems for rehabilitation," *J. NeuroEng. Rehabil.*, p. 2, Jun. 2005.

[63] U.S. Food and Drug Administration, Center for Devices and Radiological Health, Infusion Pump Improvement Initiative, White Paper, Apr. 2010.

[64] K. Venkatasubramanian, S. K. S. Gupta, R. P. Jetley, and P. L. Jones, "Interoperable medical devices," *IEEE Pulse*, vol. 1, no. 2, pp. 16–27, Sep./Oct. 2010.

[65] K. J. Vicente, K. Kada-Bekhaled, G. Hillel, A. Cassano, and B. A. Orser, "Programming errors contribute to death from patient-controlled analgesia: Case report and estimate of probability," *Can. J. Anesthesiol.*, vol. 50, no. 4, pp. 328–332, 2003.

[66] A. Wassyng, T. Maibaum, M. Lawford, and H. Bherer, "Software certification: Is there a case against safety-cases," *Proc.Workshop Modeling, Development, Verification Adaptive Comput. Syst.*, vol. 6662. Berlin, Germany: Springer-Verlag, Apr. 2010, pp. 206–227, ser. Lecture Notes in Computer Science.

[67] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic, "Alarm-net: Wireless sensor networks for assisted-living and residential monitoring," Dept. Comput. Sci., Univ. Virginia, Charlottesville, VA, Tech. Rep. CS-2006-11, 2006.

ABOUT THE AUTHORS

Insup Lee (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Wisconsin, Madison, in 1983.

He is Cecilia Fitler Moore Professor of Computer and Information Science and Director of PRECISE Center at the University of Pennsylvania, Philadelphia. He holds a secondary appointment in the Department of Electrical and Systems Engineering. His research interests include cyber-phsyical systems, real-time and embedded systems, runtime assurance and verification, formal methods and tools, trust management, and high-confidence medical systems.

Dr. Lee received the IEEE TC-RTS Outstanding Technical Achievement and Leadership Award in 2008.

John Hatcliff received the Ph.D. degree in computer science from Kansas State University, Manhattan, in 1994.

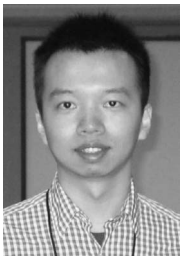
He is a University Distinguished Professor in the Computing and Information Sciences Department, Kansas State University, Manhattan. His research interests include medical device integration and coordination, verification and validation, certification and regulatory issues for medical devices, formal methods, and software engineering.

Oleg Sokolsky (Member, IEEE) received the Ph.D. degree in computer science from Stony Brook University, Stony Brook, NY, in 1996.

He is a Research Associate Professor of Computer and Information Science at the University of Pennsylvania, Philadelphia. His research interests include the application of formal methods to the development of cyber–physical systems, architecture modeling and analysis, specification-based monitoring, as well as software safety certification.

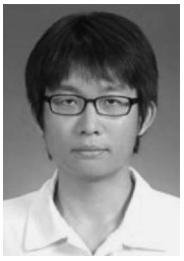
Eunkyoung Jee (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from KAIST, Daejeon, Korea, in 1999, 2001, and 2009, respectively.

She is a Research Assistant Professor in the Computer Science Department, KAIST. She was a Postdoctoral Researcher in the Computer and Information Science Department at the University of Pennsylvania, Philadelphia. Her research interest includes safety-critical software, software testing, formal method, and safety analysis.



Sanjian Chen (Student Member, IEEE) is currently working towards the Ph.D. degree in computer and information science at the University of Pennsylvania, Philadelphia.

His research interests include modeling, simulation, and formal analysis of networked cyber-physical systems, compositional design and analysis of real-time systems, and real-time virtualization on multiprocessors.



BaekGyu Kim is currently working towards the Ph.D. degree in computer science at the University of Pennsylvania, Philadelphia.

His research interests include modeling and verification of safety-critical systems, and the automated implementation of such systems through formal method. Currently, he is working on applying safety-assured model-driven engineering to PCA pump implementation as a part of generic infusion pump project.

Andrew King (Student Member, IEEE) received the B.S. and M.S. degrees in computer science from Kansas State University, Manhattan, in 2005 and 2009, respectively. He is currently working towards the Ph.D. degree in computer science at the University of Pennsylvania, Philadelphia.

His research interests include modeling, verification, and certification of distributed systems and software, in particular systems that can be integrated and reconfigured at runtime.

Margaret Mullen-Fortino is currently working towards the Ph.D. degree in health policy at the University of the Sciences, Philadelphia, PA.

She is the Operations Director for Penn e-lert eICU, the University of Pennsylvania Health System critical care telemedicine program. She is a nurse clinical consultant on the MCPS project, offering health care and nursing domain knowledge to enhance medical software development. Her research interests include informatics and telemedicine.

Soojin Park received the B.Sc. degree from Brown University, Providence, RI, in 1997, the M.D. degree from Drexel University, Philadelphia, PA, in 2002, the Residency from Boston University, Boston, MA, in 2006, and the Fellowship from Massachusetts General Hospital, Boston, MA, in 2008.

She is a Neurointensive Care Physician at the Hospital of the University of Pennsylvania, Philadelphia, where she is the Director of Neurocritical Care Monitoring and Informatics. She is an Assistant Professor of Neurology at the Perelman School of Medicine and holds secondary academic appointments in the Departments of Neurosurgery and Anesthesiology & Critical Care. Her research interests include the development of clinical decision support tools for real-time decision making in the intensive care unit.

Alexander Roederer received the B.S. degree in computer science and mathematics from the University of Miami, Miami, FL, in 2009. He is currently working towards the Ph.D. degree in computer and information science at the University of Pennsylvania, Philadelphia.

His research interests include the integration of machine learning methods and cyber–physical systems to develop medical decision support systems.

Krishna K. Venkatasubramanian (Member, IEEE) received the Ph.D. degree in computer science from Arizona State University, Tempe, in 2009.

He is a Postdoctoral Researcher with the Department of Computer and Information Science, University of Pennsylvania, Philadelphia. His research interests include secure cyber–physical systems, body area networks, trust management, and medical device security.