The final version of this chapter appears in *Cyber-Physical Systems*, Edited by Raj Rajkumar, Dionisio de Niz, and Mark Klein, Addison-Wesley, 2017

Medical Cyber-Physical Systems[1]

(Draft)

Insup Lee, Anaheed Ayoub, Sanjian Chen,

Baekgyu Kim, Andrew King, Alexander Roederer, Oleg Sokolsky

June 2016.

Medical cyber-physical systems (MCPS) are life-critical, context-aware, networked systems of medical devices that are collectively involved in treating a patient. These systems are increasingly used in hospitals to provide high-quality continuous care for patients in complex clinical scenarios.. The need to design complex MCPS that are both safe and effective has presented numerous challenges. These challenges include achieving high levels of assurance in system software, interoperability, context-aware decision support, autonomy, security and privacy, and certification. This chapter discusses these challenges in developing MCPS, case studies that illustrate these challenges and suggest ways to address them, and highlights several open research and development issues. This chapter concludes with discussion on implications of MCPS on stakeholders and practitioners.

1 Introduction and Motivation

The two most significant recent transformations in the field of medical devices are the high degree of reliance on software-defined functionality and the wide availability of network connectivity. The former transformation means that software plays the ever more significant role in the overall device safety. The latter implies that, instead of stand-alone devices that can be designed, certified, and used independently of each other to treat patients, networked medical devices will work as distributed systems that simultaneously monitor and control multiple aspects of the

[1] Research is supported in part by NSF grants CNS-1035715, IIS-1231547, ACI-1239324, and NIH grant 1U01EB012470-01.

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 (CPS). We refer to these as medical cyber-physical systems (MCPS).

The goal of MCPS is to improve the effectiveness of patient care by providing personalized treatment through sensing and patient model matching while ensuring safety. However, the increased scope and complexity of MCPS relative to traditional medical systems present numerous developmental challenges. These challenges need to be systematically addressed through the development of new design, composition, verification, and validation techniques. The need for these techniques presents new opportunities for researchers in MCPS and more broadly in general embedded and CPS systems. One of the primary concerns in MCPS development is the assurance of patient safety. We believe that new capabilities of future medical devices and new techniques for developing MCPS with these devices will require new regulatory procedures to approve their use for treating patients. 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 there is an urgent need to ease this process without compromising the level of safety it delivers.

In this chapter, we advocate a systematic analysis and design of MCPS for coping with their inherent complexity. Consequently, model-based design techniques should play a larger role in MCPS design. Models should cover not only devices and communications between them, but also, of equal importance, patients and caregivers. The use of models will allow developers to assess system properties early in the development process and build confidence in the safety and effectiveness of 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 ultimate goal is to use model-based development as the foundation for building safe and effective MCPS. Below, we describe some of the research directions that we are taking toward addressing some of the challenges involved in building MCPS.

We view MCPS in a bottom-up manner, first describing issues associated with individual devices and then progressively increasing its complexity by adding communication, intelligence, and feedback-control. The chapter is organized as follows: (1*) Stand-Alone Devic*e: model-based high assurance software development scheme is described for stand-alone medical devices such as PCA pumps and pacemakers; (2) *Device Interconnection*: a medical device interoperability framework is presented for describing, instantiating, and validating clinical interaction scenarios; (3) *Adding Intelligence:* a smart alarm system is presented that takes vital signs data from various interacting devices to inform caregivers of potential patient emergencies and non-operational issues about the devices; (4) *Automated Actuation/Delivery:* a model-based closed-loop care delivery system is presented, which can autonomously deliver care to the patients based on the current state of the patient; and (5) *Assurance Cases*: the use of assurance cases is described for organizing collections of claims, arguments, and evidence to establish the safety of a medical device system. Preliminary discussion of some of these challenges have appeared in [Lee12].

2 System Description and Operational Scenarios

An Overview of MCPS.

MCPS are *safety-critical*, *smart* systems of *interconnected* medical devices that are collectively involved in treating a patient within a specific *clinical scenario* The clinical scenario determines treatment options that can be chosed and adjustments of treatment settings that need to be made in response to changing patient state. Traditionally, decisions about the treatment options and settings are made by an attending caregiver, who makes them by monitoring patient state using individual devices and performs manual adjustments. Thus, 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." MCPS alter this view by introducing additional computational entities that aid the caregiver in controlling the "plant." Figure 1 shows the conceptual overview of MCPS. Devices used in MCPS can be categorized into two large groups based on their primary functionality: *monitoring devices*, such as bedside heart-rate and oxygen-level monitors and sensors, which provide different kinds of clinic-relevant information about patients; and *delivery devices*, such as infusion pumps and ventilators, which actuate therapy capable of changing the patient's physiological state. In MCPS, interconnected monitoring devices can feed collected data to a decision support or administrative support entities, each of which serves a different, albeit complementary, purpose. For example, 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

Most medical devices rely on software components for carrying out their tasks. Ensuring safety of the devices and their interoperation is crucial. One of the more effective ways of ensuring this is to use model-based development methods, which can ensure device safety by enabling medical device verification. This also opens the door for eventually certification of the devices to meet certain safety standards.

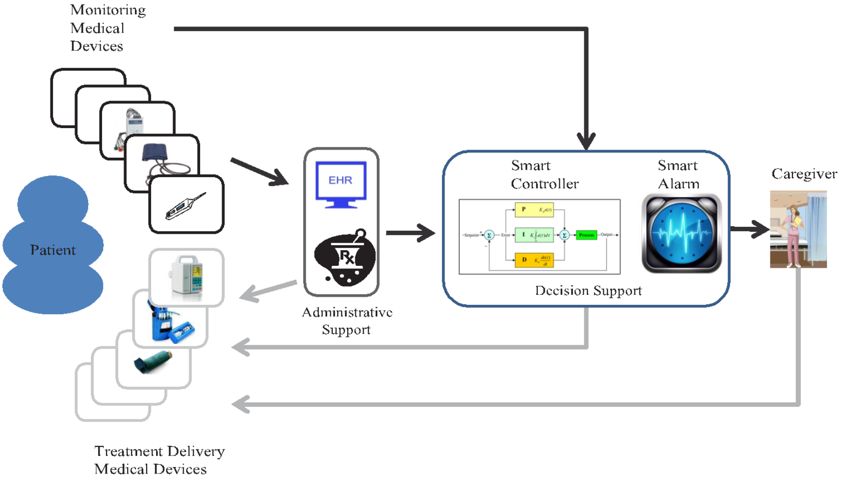


Figure 1: Medical Cyber-Physical Systems: Conceptual Overview

Virtual Medical Devices.

Given the complexity of MCPS, it has to be user-centric, i.e., easy to setup and use, in an largely automated manner. One of the ways of accomplishing this based by developing a description of the MCPS workflow and then enforcing it on physical devices. MCPS workflow can be described in terms of (1) number and types of devices involved, (2) their mutual interconnections, and (3) the clinical supervisory algorithm needed for coordination and analysis of data collected. Such description defines *Virtual Medical Device* (VMD). VMD are used by a *VMD App* and instantiated into a setup of actual medical devices called *Virtual Medical Device Instance* (VMD instance). The devices in VMD instance are usually interconnected using some form of interoperability middleware, which is responsible for ensuring that the inter-device connections are correctly configured. The principal task of the VMD App is therefore to find the medical devices in VMD instance (which may be quite large), establish network connections between them and install the clinical algorithm into the supervisor module of the middleware for managing the interactions of the clinical workflow and reasoning about the data produced.

Basically, when the VMD App is started, the supervisor reads the VMD App specification and tries to couple all involved devices accordingly. Once the workflow has run its course, the VMD App can perform the necessary cleanup in order to allow another workflow to be specified using different combination of medical devices in the VMD instance.

Clinical Scenarios.

Each VMD is to support a specific clinical scenario with a detailed description of how devices and clinical staff work together in a clinical situation or event. Here, we describe two such scenarios: one for X-ray & Ventilator coordination and another for PCA safety interlock system.

One example that illustrates how patient safety can be improved by MCPS is by developing VMD that coordinates the interaction between an X-ray machine and a ventilator. Consider the scenario taken from [Lofsky04]. X-ray images are often taken during surgical operations. If the operation is being performed under general anesthesia, the patient is breathing with the help of a ventilator. Because the patient on ventilator cannot "hold its breath" to let the X-ray image be taken without the blur caused by moving lungs, the ventilator has to be paused and later restarted. There have been cases where the ventilator was not restarted, leading to the death of the patient. Interoperation of the two devices can be used in several ways to ensure that patient safety is not compromised, as discussed in [Arney09]. One possibility is to let the X-ray machine pause and restart the ventilator automatically. A safer alternative, although presenting tighter timing constraints, is to let the ventilator transmit its internal state to the X-ray machine. There typically is enough time to take an X-ray image at the end of the breathing cycle, when the patient has finished exhaling until the start of the next inhalation. This approach requires the X-ray machine to know precisely the instance when the air flow rate becomes close enough to zero and the time when the next inhalation starts. Then, it can make the decision to take a picture if enough time – taking transmission delays into account – is available.

Another clinical scenario that can easily benefit from the closed-loop approach of MCPS is patient-controlled analgesia (PCA). PCA infusion pumps are commonly used to deliver opioids for pain management, for instance after surgery. Patients have very different reactions to the medications and require very different dosages and delivery schedules. PCA pumps give the patient a button to press to request a dose when they decide they want it rather than using a schedule fixed by a caregiver. Some patients may decide they prefer a higher level of pain to the nausea the drugs may cause and can press the button less often, while patients who need a higher dose can press it more often. A major problem with opioid medications in general is that an excessive dose can cause respiratory failure. A properly programmed PCA system should not allow an overdose because it is programmed with limits on how many doses it will deliver, regardless of how often the button is pushed. However, this safety mechanism is not sufficient to protect all patients. Some patients still receive overdoses if the pump is misprogrammed, if the pump programmer overestimates the maximum dose a patient can receive, if the wrong concentration of drug is loaded into the pump, or if someone other than the patient presses the button (PCA-by-proxy), among other causes. PCA infusion pumps are currently involved in a large number of adverse events, and existing safeguards such as drug libraries and programmable limits are not adequate to address all the scenarios seen in clinical practice [Nuckols08].

3 Key Design Drivers and Quality Attributes

While software-intensive medical devices such as infusion pumps, ventilators, and patient monitors have been used for a long time, the field of medical devices is currently undergoing a rapid transformation. The changes under way bring new challenges to the development of high-confidence medical devices, but at the same time they open new opportunities for the research community [Lee06]. This section starts with the main trends that have emerged recently, identifies quality attributes and challenges, and provide detailed discussion on several MCPS specific topics.

3.1 Trends

New software-enabled functionality.

Following the general trend in the field of embedded systems and more broadly cyber-physical systems, introduction of the new functionality is largely driven by the new possibilities that software-based development of medical device systems is offering. A prime example of the new functionality is seen in the area of robotic surgery, which requires real-time processing of high-resolution images and haptic feedback. Another example is proton therapy treatment. It is one of the most technology-intensive procedures and requires one of the largest-scale medical device systems. Used to deliver precise doses of radiation for cancer patients, the treatment requires precise guiding of the proton beam from a cyclotron to patients, requiring adaptation to even minor shifts in position. Higher precision of the treatment, compared to conventional radiation therapy, allows higher radiation doses to be applied. This, in turn, places more stringent requirements on patient safety. Control of proton beams is subject to very tight timing constraints, with much less tolerance than for most medical devices. To further complicate the problem, the same beam is applied to multiple patient locations and needs to be switched from location to location, opening up the possibility of interference between beam scheduling and beam application. In addition to the proton beam control, a highly critical function of software in a proton treatment system is real-time image processing to determine precise position of the patient and detect any patient movement. In [Rae03], the authors have analyzed the safety of proton therapy machines, however their analysis concentrates on a single system, the emergency shutdown. In general, proper analysis and validation of such large and complex systems remains one of the big challenges facing the medical device industry.

However, even in simpler devices, such as pacemakers and infusion pumps, more and more software-based features are added, making device software more complex and error-prone [Jeroeno4]. Rigorous approaches are required to make sure that software in these devices operates correctly. Because these devices are relatively simple, they are good candidates for case studies of challenges and experimental development techniques. Some of these devices, such as pacemakers, are being used as challenge problems in the formal methods research community [McMaster13].

Increased connectivity of medical devices.

In addition to relying more and more on software, medical devices are increasingly equipped with network interfaces. Interconnected medical devices, effectively, form a distributed medical device system of a larger scale and complexity that has to be properly designed and validated to ensure effectiveness and patient safety. Today, the networking capabilities of medical devices are primarily used for patient monitoring (through local connection of individual devices to integrated patient monitors or for remote monitoring in a tele-ICU [Sapirstein09] setting) and for interaction with electronic health records to store patient data.

The networking capabilities of most medical devices today are limited in functionality and tend to rely on proprietary communication protocols offered by major vendors. There is, however, a growing realization among clinical professionals that open interoperability between different medical devices will lead to improved patient safety and new treatment procedures. Medical Device Plug-and-Play (MD PnP) Interoperability initiative [Goldman05, MDPNP] is a relatively recent effort that aims to provide an open standards framework for safe and flexible interconnectivity of medical devices, in order to improve patient safety and health care efficiency. In addition to developing interoperability standards, MD PnP initiative collects and demonstrates clinical scenarios where interoperability leads to improvement over the existing practice.

Physiological closed-loop systems.

Traditionally, most clinical scenarios have a caregiver – and often more than one – controlling the process. For example, an anesthesiologist monitors sedation of a patient during an operation and decides when an action to adjust the flow of sedative needs to be taken. There is a concern in the medical community that such reliance on "human in the loop" may compromise patient safety. Caregivers, who are often overworked and operate under severe time pressure, may miss a critical warning sign. Nurses typically care for multiple patients at a time and can be distracted at a wrong moment. Using an automatic controller to provide continuous monitoring of the patient state and handling of routine situations would be a big relief to the caregiver and can improve patient care and safety. Although the computer will probably never replace the caregiver completely, it can significantly reduce the workload, calling the caregiver's attention only when something out of the ordinary happens.

Scenarios based on physiological closed-loop control have been used in the medical device industry for some time. However, their application has been mostly limited to implantable devices that cover relatively well understood body organs, such as the heart in the case of pacemakers and defibrillators. Implementing closed-loop scenarios in distributed medical device systems is a relatively new idea that has not made its way to the mainstream practice.

Continuous Monitoring and Care.

Due to a high cost associated with in-hospital care, there has been increasing interest in alternatives such as home care, assisted living, telemedicine, and sport-activity monitoring. Mobile monitoring and home monitoring of vital signs and physical activities allow health to be assessed remotely at all times. Also, there is a growing popularity of sophisticated technologies such as body sensor networks to measure training effectiveness and athletic performance based on physiological data such as heart rate, breathing rate, blood-sugar level, stress level, and skin temperature. However, most of the current systems operate in store-and-forward mode, with no real-time diagnostic capability. Physiological closed-loop technology will allow diagnostic evaluation of vital signs in real-time and make constant care possible.

3.2 Quality Attributes and Challenges of the MCPS domain

Building MCPS applications requires ensuring the following quality attributes, when in turn poses important challenges:

- *Safety*: 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. We advocate the use of model-based development and analysis as means of ensuring safety of MCPS.

- *Interoperability*: Many modern medical devices are equipped with network interfaces, enabling us to build MCPS with new capabilities by combining existing devices. Key to this is the concept of interoperability, where individual devices can exchange information facilitated by an application deployment platform. It is essential to ensure that the MCPS built from interoperable medical devices are safe, effective, secure, and can eventually be certified as such.

- *Context-Awareness*: Integration of patient information from multiple sources can provide a better understanding of the state of the patient's health, and use it to enable early detection of ailments and generation of effective alarms in the event of emergencies. However, given the complexity of human physiology and variations of physiological parameters over patient population, developing such computational intelligence is a non-trivial 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. Safety analysis of autonomous decisions in the resulting closed-loop system is a big challenge, primarily due to the complexity and variability of human physiology.

- *Security and Privacy*: Medical data collected and managed by MCPS is very sensitive. 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. Network connectivity enables new MCPS functionality through exchanging patient data from multiple sources; however, it also increases vulnerability of the system to security and privacy violations.

- *Certification*: A report by the U.S. National Academy of "Science, Software for Dependable Systems: Sufficient Evidence?," recommends evidence-based approach to the certification of high-confidence systems such as MCPS using explicit claims, evidence and expertise [Jackson07]. The complex and safety-critical nature of MCPS requires a cost-effective way to demonstrate medical device software dependability. Certification is therefore an essential requirement for the eventual viability of MCPS and an important challenge to be addressed. An assurance case is a structured argument supported by a documented body of evidence to provide a convincing and consistent argument that a system is adequately safe (or secure) [Menon09]. The notion of assurance cases hold the promise of providing an objective, evidence based approach to software certification. Assurance cases are increasingly used as a means for demonstrating safety in industries such as Nuclear Power, transportation, and automotive systems, and are mentioned in the recent IEC 62304 development standard for medical software.

3.3 High-Confidence Development of MCPS

3.3.1 Motivation

Most new functionality in medical devices is software based, and many functions traditionally implemented in hardware – including safety interlocks – are relegated to software. Thus, high-confidence software development is very important for the safety and effectiveness of MCPS.

A relatively conventional approach to high-assurance development of safety-critical systems based on the mitigation of hazards is illustrated in Figure 2. The process starts with the identification of desired functionality and hazards associated with the system operation. The chosen functionality yield system functional requirements, while hazard mitigation strategies yield system safety requirements. Functional requirements are used to build detailed behavioral models of the software modules, while safety requirements are turned into properties that these models should satisfy. Models and their desired properties are the inputs to the model-based software development, which is comprised of verification, code generation, and validation phases.

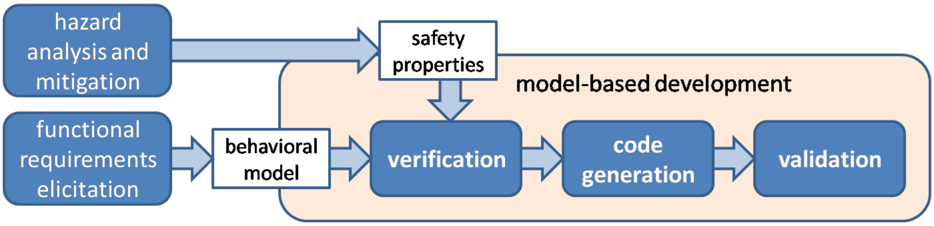


Figure 2: High-assurance development process for embedded software

Model-based development has emerged as a means of raising the level of assurance in software system. In this approach, developers start with declarative models of the system perform rigorous model verification with respect to safety and functional requirements, and then use systematic code generation techniques to 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 semi-formal approaches such as UML and Simulink [Becker09] and thus do not allow developers to fully utilize the benefits of model-based design. The use of formal modeling enables making mathematically sound conclusions about the models and generating code from them.

3.3.2 Challenges

There are several challenges in developing the MCPS through the model-driven implementation process. The first challenge is choosing the right level of abstraction for the modeling effort. A highly abstract model makes the verification step relatively easy to perform. However a model that is too abstract is difficult to use in the code generation process, since too many implementation decisions have to be guessed by the code generator. On the other hand, a very detailed model makes code generation relatively straightforward, however, such models push the limits of available verification tools.

Many modeling approaches rely on the separation of platform-independent and platform-dependent aspects. From the modeling and verification perspective, there are several reasons to separate the platform-independent aspects from the platform-dependent aspects.

First, hiding platform-dependent details reduces the modeling and verification complexity. Consider, for example, the interaction between a device and its sensors. For code generation, one may need to specify the details of how the device retrieves data from sensors. A sampling-based mechanism with a particular sampling interval will yield a very different generated code, compared to an interrupt based mechanism. However, exposing such details in the model adds an additional level of complexity to the model, complicating verification time increases too much.

In addition, abstracting away from a particular platform allows us to use the model across different target platforms. Different platforms may have different kinds of sensors that supply the same value. For example, consider an empty-reservoir alarm that many infusion pumps implement. Some pumps may not have a physical sensor for that purpose and estimate the remaining amount of medication based on the infusion rate and elapsed time. Other pumps may have a sensor based on syringe position or pressure in the tube. Abstracting away these details would allow us to implement the same pump control code on different pump hardware. However, such separation leads to the integration challenges at the implementation level. The generated code from the platform-independent model needs to be integrated with different target platforms in a way that preserves the verified properties of the platform-independent model.

Second, there is often a semantic gap between the model and implementation. A system is modeled using the formal semantics provided by the chosen modeling language. However, some of the model semantics may not match well with that of implementation. For example, in UPPAAL and Stateflow, the interaction between the PCA pump and the environment (e.g, user or pump hardware) can be modeled using instantaneous channel synchronization or event-broadcasting that takes zero-time delay. Such semantics simplifies modeling input and output of the system so that the modeling/verification complexity is reduced. However, the correct implementation of such semantics is hardly realizable at the implementation level since execution of those actions requires interaction among components that take non-zero time-delay.

A case study presented below concentrates on developing a PCA (Patient-Controlled Analgesic) infusion pump system and considers several approaches to address these challenges.

3.3.3 Case study

PCA infusion pumps.

A Patient-Controlled Analgesic (PCA) infusion pump is a type of infusion pump that primarily delivers pain relievers, and is equipped with a feature that allows for additional limited delivery of medication, called bolus, upon patient demand. This type of infusion pumps are widely used for pain control of post-operative patients. In case the pump overdoses such opioid drugs, the patient can be at risk of respiratory depression and death. Therefore, it is subject to stringent safety requirements that aim to prevent overdose.

According to FDA's Infusion Pump Improvement Initiative [FDA10a], the FDA has received over 56,000 reports of adverse events associated with the use of infusion pumps from 2005 through 2009. In the same period, 87 recalls of infusion pumps were conducted by the FDA, affecting all major pump manufacturers. The prevalence of the problems clearly indicates the need for better development techniques.

The GPCA Project.

The Generic PCA project, a joint effort between PRECISE Center at the University of Pennsylvania and researchers at the U.S. Food and Drug Administration, 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 [UPenn-b], a set of safety requirements [UPenn-a], and a reference model of PCA infusion pump systems [UPenn]. Based on these documents, one can develop PCA infusion pump controller software following a model-driven implementation.

In the case study, software for the PCA pump controller is developed by using the model-driven implementation approach starting from the reference model and the safety requirements. A detailed account of the effort is presented in [Kim11].

The development approach follows the process outlined in Figure 2. The detailed steps are shown in Figure 3. In addition, the case study included the construction of an assurance case, a structured argument based on the evidence collected during the development process, which aims to convince evaluators that the GPCA-reference implementation complies to its safety requirements. The assurance case development is discussed in more detail in Section 3.7.

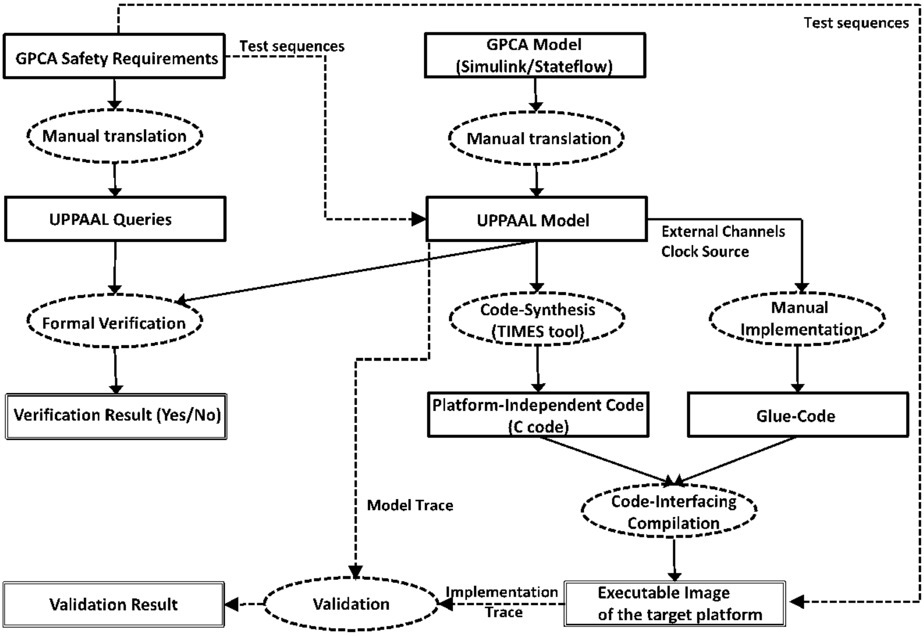


Figure 3: The Model-Driven Development for the GPCA prototype.

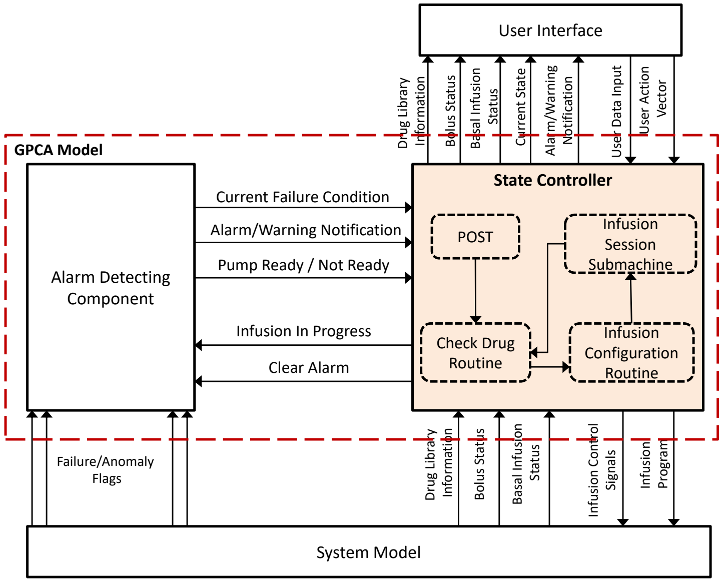


Figure 4: The System Architecture of the GPCA Model.

Modeling.

The reference model of the GPCA pump implemented in Simulink/STATEFLOW is used as the source of functional requirements and converted in via a manual but systematic translation to UPPAAL [Behrmann04]. The model structure follows the overall architecture of the reference model, which is shown in Figure 4. The software is organized into two state machines, the state controller and the alarm detecting component. The user interface has been considered in a follow-up case study [Masci13]. Both state machines interact with sensors and actuators on the pump platform.

The state machines are organized as a set of modes, with each mode captured as a separate sub-machine. In particular, the state controller contains four modes: (1) POST (Power-On Self Test) mode is the initial mode that checks system components on start-up; (2) The check-drug mode represents a series of checks that the caregiver performs to validate the drug loaded into the pump; (3) The infusion configuration mode represents interactions with the caregiver to configure infusion parameters such as infusion rate and VTBI (volume to be infused) and validate them against the limits encoded in the drug library; (4) the infusion session, where the pump controls delivery of the drug according to the configuration and patient's bolus requests.

Model verification.

GPCA safety requirements are expressed in English as "shall" statements. Representative requirements are "*No normal bolus doses shall be administered when the pump is alarming*" and “*The pump shall issue an alert if paused for more than t minutes*".

Before verification could be performed, requirements need to be formalized as properties to be checked. One can categorize the requirements according to their precision and level of abstraction: (A) requirements that are detailed enough to be formalized and verified on the model; (B) requirements that are beyond the scope of the model; (C) requirements that are too imprecise to be formalized. Only requirements in category A can be readily used in verification, however, just 20 out of 97 GPCA requirements fell into this category.

Most of the requirements in Category B concern the functional aspects of the system that are abstracted away at the modeling level. For example, consider the requirement "*If the suspend occurs due to a fault condition, the pump shall be stopped immediately without completing the current pump stroke*". There is another requirement to complete the current stroke under other kinds of alarms. That is, the motor needs to be stopped in different ways in different circumstances. These requirements fall into Category B, since the model does not detail the behavior of the pump stroke. Handling of properties in this category can be done in several ways.

One way is to introduce additional platform-specific details into the model, increasing complexity of the model. However, this would blur the distinction between platform-independent and platform-specific models, which is useful in the model-based development. An alternative approach is to handle these requirements outside of the model-based process, e.g., validating by testing. In this case, however, the benefits of formal modeling are lost.

A better approach is to match the levels of detail by further decomposing the requirements. At the platform-independent level, one can check that the property that the system performs two different stop actions in response to different alarm conditions (which would be a Category A requirement). Then, at the platform-specific level, one needs to check that one stop action corresponds to immediate stopping of the motor, while the other stop action lets the motor complete the current stroke.

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 is a simple example of a deficiency in the requirement specification uncovered during formalization.

Once the categorization of the requirements is complete, requirements in Category A are formalized and verified using a model checker. In the case study, the requirements were converted into UPPAAL queries. Queries in UPPAAL use a subset of Timed CTL temporal logic and can be verified using the UPPAAL model checker.

Code generation and system integration.

Once the model is verified, a code generation tool is used to produce the code in property-reserving manner. An example of such tool is TIMES [Amnell03] for UPPAAL timed automata. Since the model is platform-independent, the resulting code is also platform-independent. For example, the model does not specify how the actual infusion pump interacts with sensors and actuators attached to the specific target platform. Input and output actions (e.g., a bolus request by a patient or an occurrence of the occlusion alarm from the pump hardware) are abstracted as instantaneous transitions associated with input/output synchronization with the their environment. On a particular platform, the underlying operating system is scheduling the interactions, affecting the timing of the execution.

There are several approaches to address this issue at the integration stage. In [Henzinger07], the higher-level programming abstraction is proposed so that timing aspects can be modeled and generated into code that is independent from scheduling algorithms of a particular platform. Then, the platform integration is performed by verifying time-safety to check if the platform-independent code can be scheduled on the particular platform. Another approach is to systematically generate an I/O interface that helps the platform-independent and dependent code to be integrated in a traceable manner [Kim12]. From the code generation perspective, [Lublinerman09] proposed a way to generate code for a given composite block of the model independently from context and using minimal information about the internals of the block.

Validation of the implementation.

Unless the operation of an actual platform is completely formalized, there will invariably be assumptions made during the verification and code generation phases that cannot be formally guaranteed. The validation phase is meant to check that these assumptions do not break the behavior of the implementation. In the case study, a test harness that systematically exercises the code using test cases derived from the model. There is a rich literature on model-based test generation, see [Dias07] for a survey of the area. The goal of such testing-based validation is to systematically detect deviations of the system behavior from that of the verified model.

3.3.4 Remarks and Discussion

Extreme market pressures faced by the medical device industry force companies to reduce development cycles as much as possible. The challenge is to find a development process that will deliver a high degree of safety assurance under these conditions. We believe that model-based development can be a significant part of such development process. The case study discussed in this section illustrates the steps of the high-assurance development process using a simple medical device. Each of the steps can be implemented in a variety of ways. The choice of modeling, verification, and code generation technologies depends on factors such as complexity and criticality level of the application. However, the process itself is general enough to accommodate a wide variety of rigorous development technologies.

3.4 On-Demand Medical Devices and Assured Safety

Historically, medical devices have been used as individual tools for patient therapy. In order to provide complex therapy caregivers (i.e., physicians and nurses) must coordinate the activities of the various medical devices manually. This is burdensome for the caregiver, error and accident prone.

One example of manual device coordination in current practice is the X-ray & Ventilator coordination mentioned in Section 2 and another example is trachea or larynx surgery performed with a laser scalpel. In this type of surgery, the patient is under general anesthesia while the surgeon makes cuts on the throat using a high intensity laser. Because the patient is under anesthesia, their breathing is supported by an anesthesia ventilator which supplies a high concentration of oxygen to the patient. This situation presents a serious hazard: if the surgeon accidentally cuts into the breathing tube using the laser, the increased concentration of oxygen can lead to a rapid combustion, burning the patient from the inside out. In order to mitigate this hazard, the surgeon and anesthesiologist must constantly communicate: When the surgeon needs to cut, he or she signals the anesthesiologist who reduces or stops the oxygen being supplied to the patient. If the patient's oxygenation levels drop too low, the anesthesiologist will signal the surgeon to stop cutting so oxygen can be supplied again.

If medical devices could coordinate with one another, then the surgeon and anesthesiologist would not have to expend concentration and effort to ensure that the activity of the medical devices are safely synchronized. Furthermore, the patient would not be exposed to the potential for human error. There are many clinical scenarios which would benefit from automated medical device coordination. These scenarios involve at least one of *device synchronization*, *data fusion*, or *closed-loop control*. The laser scalpel ventilator safety interlock epitomizes device synchronization: each device must always been in a correct state relative to other devices. In data-fusion physiologic readings from multiple separate devices are considered together. Examples of such applications include smart alarms and clinical decision support systems (see Section 3.5). Additionally, closed-loop control of therapy can be achieved by collecting data from devices that sense patient's physiological state and then using that data to control actuators such as infusion pumps (see Section 3.6).

3.4.1 Definition - Virtual Medical Devices

A collection of devices working in unison to implement a given clinical scenario is, in essence, a new medical device. Such collections have been referred to as virtual medical devices (VMDs) because no single manufacturer is producing this device and delivering it fully formed to the clinician. A VMD does not exist until assembled at the patient's bedside. A VMD instance is created each time the clinician assembles a set of devices for the VMD and connects them together.

3.4.2 Challenges

There are several existing standards designed to enable medical device interconnectivity and interoperability. These standards include the Health Level 7 standards [Dolin06], IEEE-11073 [Iso/ieee11073, Clarke07], and the IHE profiles [Carr03]. While these standards enable medical devices to exchange and interpret data, they do not adequately address more complex interactions between medical devices such as inter-device coordination and control such as with the laser scalpel and ventilator. The notion of a VMD poses one major fundamental question: How does one assure safety in systems that are assembled by their users? Traditionally, most safety-critical cyber-physical systems, such as aircraft, nuclear power plants, and medical devices, are evaluated for safety by regulators before they can be used. The state of the art in safety assessment is to consider the complete system. This is possible because the complete system is manufactured by a single systems integrator. However, as mentioned before, virtual medical devices are constructed at bedside, based on the needs of an individual patient and from available devices. This means that a caregiver may instantiate a VMD from a combination of medical devices (i.e., varying in terms of make, model, feature set) that have never been combined into an integrated system for that particular clinical scenario. Finally, "on-demand" instantiation of the VMD confounds the regulatory pathways for medical devices that are currently available. In particular, there is no consensus on the role of the regulator when it comes to VMD. Should regulators mandate specific standards? Do regulators need to adopt component wise certification regimes? What is the role, if any, of third part certifiers?

3.4.3 Case Studies - The Integrated Clinical Environment (ICE) and Medical Device Coordination Framework (MDCF)

The subject of safety assessment of on-demand medical systems has been the focus of a number of research projects. These projects have explored different aspects of on-demand medical systems, their safety, and possible mechanisms for regulatory oversight. The Medical Device Plug & Play project articulated the need for on-demand medical systems, documented specific clinical scenarios that would benefit, and developed the Integrated Clinical Environment (ICE) architecture, which has been codified as an ASTM standard (ASTM F2761-2009) [ASTM09]. ICE proposes to approach the engineering and regulatory challenges by building medical systems around a system architecture that supports compositional certification. In such an architecture (Figure 5), each medical system would be composed out of a variety of components (clinical applications, a medical application platform, and medical devices), which would be regulated, certified, and then obtained by the health-care organization separately [Hatcliff12].

ICE.

Figure 5 shows the primary components of the ICE architecture. The rest of this section summarizes the intended functionality and goals for each of these components. It is important to note that ASTM F2761-2009 does not provide detailed requirements for these as it is purely an architectural standard. However the roles of each of the components in the architecture imply certain informal requirements:

- *Apps*. Applications are software programs that provide the coordination algorithm for a specific clinical scenario (i.e., smart alarms, closed-loop control of devices, etc.). In addition to executable code, these applications contain device requirements declarations: a description of the medical devices they need to operate correctly. These apps would be validated and verified against their requirements specification before they are marketed.

- *Devices*. Symmetrical to the applications, medical devices would implement an interoperability standard and carry a self-descriptive model, known as a capabilities specification. Each medical device would be certified that it conforms to its specification before it is marketed and sold to end users.

- *Supervisor*. The supervisor provides a secure isolation kernel and virtual machine (VM) execution environment for clinical applications. It would be responsible for ensuring that apps are partitioned in both data and time from each other.

- *Network Controller*. The network controller is the primary conduit for physiologic signal data streams and device control messages. The network controller would be responsible for maintaining a list of connected devices and ensuring proper quality of service guarantees in terms of time and data partitioning of data streams, as well as security services for device authentication and data encryption.

- *ICE Interface Description Language*. The description language is the primary mechanism for ICE-compliant devices to export their capabilities to the network controller. These capabilities may include what sensors and actuators are present on the device, and the command set it supports.

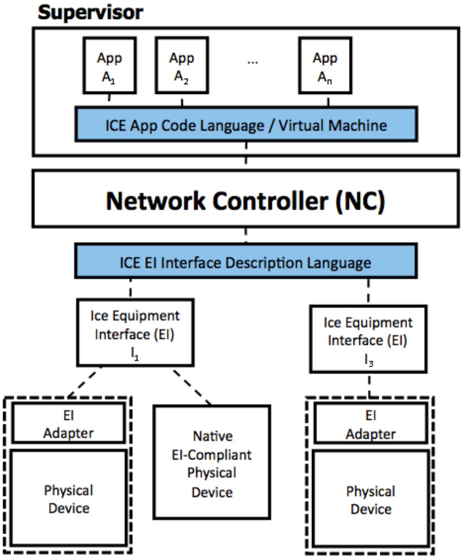

Supervisor
App A_1
App A_2
...
App A_n
ICE App Code Language / Virtual Machine
Network Controller (NC)
ICE EI Interface Description Language
Ice Equipment Interface (EI) I_1
Ice Equipment Interface (EI) I_3
EI Adapter
Physical Device
Native EI-Compliant Physical Device
EI Adapter
Physical Device

Figure 5: ICE Architecture

MDCF.

The Medical Device Coordination Framework (MDCF) [King09, MDCF] is an open-source project that aims to provide a software implementation of a medical application platform that conforms to the ICE standard. The purpose of the MDCF is to provide a modular framework that enables researchers to rapidly prototype systems and explore implementation and engineering issues associated with on-demand medical systems.

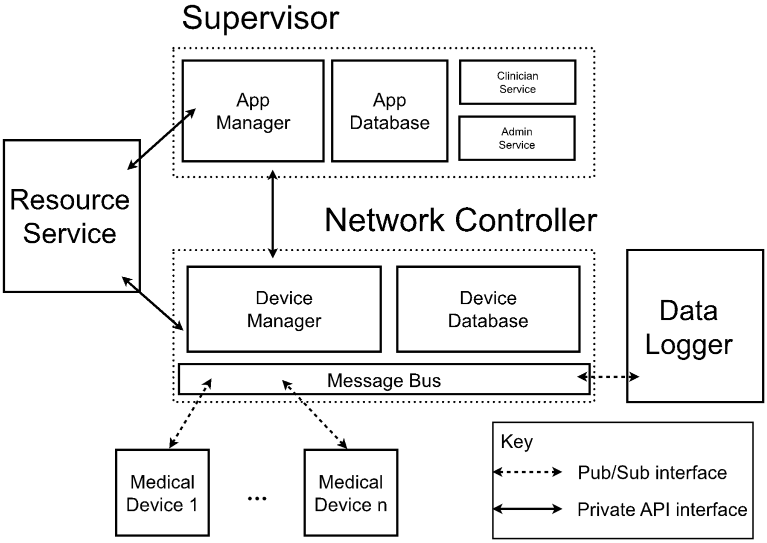


Figure 6: MDCF services decomposed along ICE architectural boundaries

The MDCF is implemented as a collection of services which work together to provide some of the capabilities required by ICE as essential for a medical application platform. The functionality of these services also decompose along the architectural boundaries defined in the ICE architecture (see Figure 6), thus the MDCF consists of "network controller" services, "supervisor" services and a global resource management service.

Network controller services are as follows:

- *Message Bus*. Abstracts the low level networking implementation (e.g., TCP/IP) and provides a publish/subscribe messaging service. All communication between medical devices and the MDCF occurs via the message bus, including protocol control messages, patient physiologic data, and commands sent from apps to devices. The Message Bus also provides basic real-time guarantees (e.g., bounded end-to-end message transmission delays) that apps can take as assumptions. Additionally, the Message Bus supports various fine-grained message and stream access control and isolation policies. While the current implementation of the message bus encodes messages using XML, the actual encoding strategy is abstracted away from the apps and devices by the message bus API which exposes messages as structured objects in memory.

- *Device Manager*. Maintains a registry of all medical devices currently connected with the MDCF. The Device Manager implements the server side of the MDCF device connection protocol (medical devices implement the client side) and tracks the connectivity of those devices, notifying the appropriate apps if a device goes offline unexpectedly. The Device Manager serves another important role: it validates the trustworthiness of any connecting device by determining if the connecting device has a valid certificate.

- *Device Database*. Maintains a list of all specific medical devices that the healthcare provider's bioengineering staff has approved for use. In particular, the database lists each allowed device's unique identifier (like an Ethernet MAC address), the manufacturer of the device, and any security keys or certificates that the Device Manager will use to authenticate connecting devices against.

- *Data Logger*. Taps into the flows of messages moving across the message bus and selectively logs them. The logger can be configured with a policy specifying which messages should be recorded. Because the message bus carries every message in the system, the logger can be configured to record any message or event that propagates through the MDCF. Logs must be tamper resistant, tamper evident, and access to logs must itself be logged, and be physically and electronically controlled by a security policy.

Supervisor Services are as follows:

- *Application Manager*. Provides a virtual machine for apps to execute in. In addition to simply executing program code, the Application Manager checks that the MDCF can guarantee the app's requirements at runtime and provides resource and data isolation, as well as access control and other security services. If the app requires a certain medical device, communications latency, or response time from app tasks but the MDCF cannot currently make those guarantees (e.g., due to system load or the appropriate medical device has not been connected), then the App Manager will not let the clinician start the app in question. If the resources are available, the application manager will reserve those resources in order to guarantee the required performance to the app. The application manager further detects and flags potential medically meaningful app interactions, since individual apps are isolated and may not be aware what other apps are associated with a given patient.

- *Application Database*. Stores the applications installed in the MDCF. Each application contains executable code and requirement metadata used by the application manager to allocate the appropriate resources for app execution.

- *Clinician Service*. Provides an interface for the clinician console GUI to check the status of the system, start apps, and display app graphical user interface elements. Since this interface is exposed as a service, the clinician console can be run locally (on the same machine) that is running the supervisor, or remotely (e.g., at a nurse's station).

- *Administrator Service*. Provides an interface for the administrator's console. System administrators can use the administrator's console to install new applications, remove applications, add devices to the device database and monitor the performance of the system.

3.4.4 Remarks

On-demand medical systems represent a new paradigm for safety-critical systems: the final system is assembled by the user instead of the manufacturer. Research into the safety assessment of these systems is an active topic. The projects described in this section represent a first step towards understanding the engineering and regulatory challenges associated with such systems. The success and safety of these systems will not only depend on new engineering techniques, but also new approaches to regulation and a willingness in industry to adopt appropriate interoperability standards.

3.5 Smart Alarms and Clinical Decision Support Systems

3.5.1 Motivation: The Noisy Intensive Care Environment

Hospital Intensive Care Units (ICUs) utilize a wide array of medical devices to care for patients. A subset of these medical devices acts as sensors which detect the intensity of various physical and chemical signals in the body. These sensors allow clinicians (doctors, nurses, and other clinical caretakers) to better understand the patient's current state. Examples of such sensors include automatic blood pressure cuffs, thermometers, heart rate monitors, pulse oximeters, electroencephalogram meters, automatic glucometers, electrocardiogram meters, etc. These sensors range from very simple to very complex. Additionally, along with traditional techniques, digital technologies have enabled new sensors to be developed and evaluated for clinical use.

The vast majority of these medical devices acts in isolation, reading a particular signal, and outputting the result of that signal to some form of visualization technology so it may be accessed by clinicians. Some devices stream data to a centralized visualization system (such as a bedside monitor or nursing station [Phillips10, Harris13]) for ease of use. However, each of the signals is still displayed independently. It is up to the clinician to synthesize the presented information to determine the patient's state.

Many of these devices can be configured to alert clinicians to a deterioration in patient state. Most devices currently in use can only be configured with threshold alarms, which activate when the particular vital sign being measured crosses a predefined threshold. While threshold alarms can be vital in the timely detection of emergency states, they have been shown to be not scientifically derived [Lynn11] and have a high rate of false alarms [Clinical07], which can be caused by insignificant random fluctuations in the patient's vital signs or noise caused by external stimuli (the most common example is patient movement, which can cause sensors to move, get compressed, or fall off). This large number of erroneous alarms causes alarm fatigue, a desensitization to the presence of these alarms which causes clinicians to ignore them [Commission13]. Alternately, in an effort to reduce the number of alarms, clinicians may improperly readjust settings on the monitor or turn off alarms entirely [Edworthy06]. Both of these can lead to miss true alarms and a decrease in quality of care [Clinical07, Donchin02, Imhoff06]. Various efforts have been made to reduce alarm fatigue. These usually focus on improving workflow, establishing appropriate patient-customized thresholds, and identifying situations where alarms are not clinically relevant [Clifford09, EBMWG92, Oberli99, Shortliffe79]. However, isolated threshold alarms cannot capture sufficient nuance in patient state to completely eliminate false alarms. 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.

Clinicians most often use multiple vital signs in concert to understand the patient's state. For example, a low heart rate (bradycardia) can be normal and healthy. However, if a low heart rate occurs in conjunction with an abnormal blood pressure or a low blood oxygen level, this can be cause for concern. Thus, it seems pertinent to develop smart alarm systems, systems that consider multiple vital signs in concert before raising an alarm. This would reduce false alarms, improving the alarm precision and reducing alarm fatigue, leading to improved care. Such a smart alarm system would be a simple version of what is in general known as a Clinical Decision Support system (CDS system) [Garg05]. Clinical decision support systems combine multiple sources of patient information with preexisting health knowledge to help clinicians make more informed decisions. It has repeatedly been shown that well designed clinical decision support systems have the potential to dramatically improve patient care, not just by reducing alarm fatigue, but by allowing clinicians to better utilize data to assess patient state.

3.5.2 Definition: Clinical Decision Support Systems

Fundamentally, CDS systems are a specialized form of MCPS with physical actuation limited to visualization. They take as inputs multiple data streams, such as vital signs, lab values, and patient history, subject them to some form of analysis, and output the results of that analysis to a clinician. A smart alarm is the simplest form of decision support system, in which multiple data streams are analyzed to produce a single alarm for the clinician. More complex systems may use trending, signal analysis, online statistical analysis, or previously constructed patient models, and may produce detailed visualizations.

3.5.3 Challenges

As CDS systems are a specialized form of MCPS, the development of CDS systems thus requires satisfying the core features of cyber-physical system development. In fact, without these features, CDS system development is impossible. The current lack of widespread use of CDS systems is in part due to the difficulty that has been encountered in establishing these features in a hospital setting.

One of the most fundamental of these requirements is the achievement of device interoperability. Even the simplest CDS system (such as a smart alarm system) must obtain access to real-time vital sign data being collected by a number of different medical devices attached to the patient. To obtain this data, the devices collecting the required vital signs must be able to inter-operate, if not with each other, then with a central data repository. At this repository, data could be collected, time-synchronized, analyzed, and visualized.

However, achieving interoperability in medical devices has previously been a major hurtle. Due to increased costs, the exponential blowup in regulatory difficulty, and the lucrative potential of selling a suite of devices with limited interoperability, individual device manufacturers currently have little incentive to make their devices inter-operate. Development of an inter-operable platform for device communication would enable MCPS to stream real-time medical information from different devices.

Many other challenges exist. For example, the safety and effectiveness of CDS systems is dependent on other factors, such as network reliability and real-time guarantees on message delivery. As networks in current hospital systems are often ad-hoc, highly complex, and built over many decades, such reliability is rare.

Another challenge is that of data storage. To achieve high accuracy, the parameters of the computational intelligence at the heart of a CDS system must often be tuned using large quantities of retrospective data. Dealing with big data is thus a vital component of the development of CDS systems. Addressing this problem will require hospitals to recognize the value in capturing and storing patients' data, along with adoption of dedicated hospital infrastructure to store and access data as part of routine workflow.

CDS systems require some level of context-aware computational intelligence. Information from multiple medical device data streams must be extracted and filtered, and used in concert with a patient model to create a context-aware clinical picture of the patient. There are three major ways in which context-aware computational intelligence can be achieved: by encoding hospital guidelines, by capturing clinician mental models, or by learning models statistically through machine learning on medical data.

While the majority of hospital guidelines can usually be encoded as a series of simple rules, they are often vague and/or incomplete, so while they may serve as a useful baseline, they are often insufficient on their own. Capturing clinician mental models involves interviewing a large number of clinicians about their decision making process to hand-build an algorithm. This process can be laborious, clinician thinking can be difficult to quantify in software, and the results from different clinicians can be difficult to reconcile. Creating models using machine learning is often the most straightforward approach. However, training such models requires large amounts of retrospective patient data and clear outcome labels, both of which can be difficult to acquire. When such data is available, it is often noisy, and filled with missing values. Choice of learning technique can be a difficult question, and while algorithm transparency is a good metric (to empower clinicians to understand the underlying process and avoid opaque black-box algorithms) there is no single choice of learning technique that is most appropriate for all scenarios.

3.5.4 Case Study: A Smart Alarm system for post-CABG surgery patients

Post-operative Coronary Artery Bypass Graft (CABG) patients are at particular risk of physiologic instability. Thus these patients are routinely subject to continuous monitoring of a combination of common vital signs. The hope is that detection of physiologic changes will allow practitioners to intervene in a timely manner and prevent post-surgery complications. As previously discussed, however, these continuous vital sign monitors are equipped only with simple threshold-based alarms, which, coupled with the rapidly-evolving post-surgical state, can lead to a large number of erroneous false positive alarms. For example, it is common for the finger clip sensors employed by pulse oximeters to fall of the patients as they get situated in their ICU bed, or for changes in the artificial lighting of the care environment to produce erroneous readings.

To reduce these and other erroneous alarms, a smart alarm system was developed which combines four main vital signs routinely collected in the Surgical ICS (SICU): blood pressure (BP), heart rate (HR), respiratory rate (RR) and blood oxygen saturation (SpO2). ICU nurses were interviewed to determine appropriate ranges for binning each vital sign into a number of ordinal sets (e.g., "Low," "Normal," "High," "Very High", leading to classifying, for example, a blood pressure above 107 as "High"). Binning vital signs in this way helped overcome the difficulty of establishing a ruleset customized to each patient's baseline vital signs. The binning criteria can be modified to address a specific patient with, for example, a very low "Normal" resting heart rate, without rewriting the entire rule set.

Afterward, a set of rules were developed in conjunction with these nurses to identify combinations of these vital sign statuses which would be cause for concern. The "smart" alarm monitors a patient's four vitals, categorizes which ordinal set they belong to, and searches the rule table for the corresponding alarm level to output. To deal with missing data (due to network or sensor faults), rapid drops to zero are conservatively classified as "Low" for the duration of the signal drop.

This smart alarm avoided many of the challenges that normally face CDS systems in the clinical environment. The set of vital signs employed was very limited and included only vital signs which are commonly collected and synchronized by the same medical device. As the "intelligence" of the smart alarm system was a simple rule table based on clinician mental models, it did not require large amounts of retrospective data to calibrate, and was transparent and easy for clinicians to understand. While network reliability would be a concern for such a system running in the ICU, the classification of missing values as "Low" provides a conservative fallback in case of a brief network failure. Additionally, running the system on a real-time middleware would provide the necessary data delivery guarantees to ensure system safety.

To evaluate the performance of the system, 27 CABG patients were observed while they convalesced in the ICU immediately after the CABG procedure. Of these 27 patients, nine had the requisite vital sign samples stored in the hospital IT system during the time period of the observation. Each of these patients was observed for between 26 and 127 minutes, totaling 751 minutes of observation. In order to compare monitor alarm performance with the CABG smart alarm, the minute by minute samples of these patients physiologic state were retroactively retrieved (after the observations) from the UPHS datastore. The smart alarm algorithm was applied to the retrieved data streams, resulting in a trace of the smart alarm outputs that would have been produced if the smart alarm were active at the patient's bed side.

Because of the relatively slow rate at which a patient can deteriorate and the expected response time of the care staff, an intervention alarm was considered to be covered by a smart alarm if the alarm occurred within 10 minutes of the intervention.

Overall, the smart alarm system produced fewer alarms. During the study, the smart alarm was active 55% of the time that the standard monitor alarms were active, and of the ten interventions during the observation time period, nine were covered by the smart alarm. The significant alarm was likely deemed significant not due to the absolute values of the vital signs being observed, but by their trend. An improved version of this smart alarm system would thus include rules concerning the trend of each of the vital signs.

3.5.5 Remarks

As more medical devices become capable of recording continuous vital sign systems, and as medical systems become increasingly interoperable, CDS systems will become essential tools to allow clinicians to process, interpret, and analyze patient data. While there are challenges facing widespread adoption of CDS systems in clinical environments, beginning to build these systems will expose their clinical utility and provide impetus for overcoming said challenges.

3.6 Closed-loop System

3.6.1 Background/Motivation

A clinical scenario can be viewed as a control loop: the patient is the plant, the controller collects information from sensors (e.g., bedside monitors) and sends configuration commands to actuators (e.g., infusion pumps) [Lee12]. Traditionally, caregivers act as the controller in most scenarios, which imposes significant decision making burden on them as one caregiver is usually caring for several patients and can check on each patient only sporadically. Continuous monitoring, where patient condition is under constant supervision, is an active area of research [Maddox08]. However, to improve patient safety further, the system should be able to also continuously react to changes in patient condition.

The smart alarm systems and decision support systems, discussed in the previous section, facilitate the integration and interpretation of clinical information, helping caregivers make decision more efficiently. Closed-loop systems aim to achieve a higher level of intelligence: in such systems, a software-based controller automatically collects and interprets physiological data, and controls the therapeutic delivery devices. Many safety critical systems utilize automatic controllers, e.g., autopilots in airplanes and adaptive cruise control in vehicles. In patient care, the controller can continuously monitor the patient's states and automatically reconfigure the actuators when the patient's condition stays within a pre-defined operation region. It will alert and hand over the control back to caregivers if a patient's state starts diverting from the safe range. Physiological closed-loop systems can take part of caregivers' workload so they can better focus on handling critical events, which would ultimately improve patient safety. In addition, software controllers can run advanced decision making algorithms (e.g., model-predictive control in blood glucose regulation [Hovorka04]) that are too computationally complicated for human caregivers, and this may improve both safety and effectiveness of patient care.

The concept of closed-loop control has been introduced to medical applications, e.g., implantable devices such as cardioverter defibrillators and other special-purpose standalone devices. A physiological closed-loop system can also be built by networking multiple existing devices, such as infusion pumps and vital sign monitors. The networked physiological closed-loop system can be modeled as a VMD.

3.6.2 Challenges

The networked closed-loop setting introduces new hazards that could compromise patient safety. The hazards need to be identified and mitigated in a systematic way. Closed-loop MCPS raise several unique challenges for safety engineering.

First, the plant, i.e., the patient, is an extremely complex system that usually exhibits significant variability and uncertainty. Physiological modeling has been a decade long challenge for bio-medical engineers and medical experts, and the area is still at the frontier of science. Unlike in many other engineering disciplines such as mechanical engineering or electronic circuit design, where high-fidelity first-principle models are usually directly applicable to theoretical controller design, the physiological models are usually non-linear and contain parameters that are highly individual-dependent, time-varying, and not easily identifiable given the technologies available. This imposes a major challenge on control design as well as system level safety reasoning.

Second, in the closed-loop medical device system, there is a complex interaction between the continuous physiology of the patient and the discrete behavior of the control software and network. Since most closed-loop systems require supervision from users (either caregivers or patients themselves), the human behavior has to be considered in the safety arguments.

Third, the control loop is subject to uncertainties caused by sensors, actuators, and communication network. For example, some body sensors are very sensitive to patient movements; vital sign monitors may alert faulty readings due to a dropped finger-clip; due to technological constraints, some bio-sensors have non-negligible error even when they are used correctly, e.g., the continuous glucose monitor [Ginsberg09]. The network behavior also has a critical impact on patient safety: patients can be harmed by the actuators if packets that carry critical control commands are dropped in the network.

3.6.3 Case Study

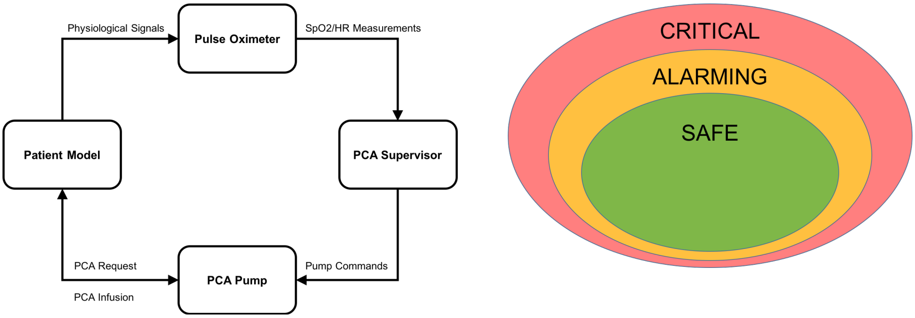
One way to systematically address the challenges listed above is to consider a model-based approach similar to the one outlined in Section 3.3, extending the high-confidence approach based on hazard identification and mitigation from individual devices to the system composed of a collection of devices and a patient.

This section briefly describes a case study of the use of physiological closed loop in pain control using a patient-controlled analgesia (PCA) infusion pump, introduced in Section 3.3.3. The biggest safety concern about the use of PCA pumps for pain control is that an overdose of an opioid analgesic can cause respiratory failure. Existing safety mechanisms built into PCA pumps include limits on bolus amounts, which are programmed by a caregiver before the start of the infusion, and minimum time intervals between consecutive bolus doses. In addition, nursing manuals prescribe periodic checks of the patient condition by a nurse. However, these mechanisms are considered insufficient to cover all possible scenarios [Nuckols08].

The case study [Pajic12] presents a safety interlock design for PCA infusion, implemented as an on-demand MCPS as described in Section 3.4 and illustrated in Figure 3.6.3. The pulse oximeter continuously monitors heart rate (HR) and blood oxygen saturation (SpO2). The controller receives measurements from the pulse oximeter and it may stop the PCA infusion if the HR/SpO2 readings indicate a dangerous decrease in respiratory activity, thereby preventing overdosing.

Safety requirements for the system are based on two regions in the space of possible patient states as reported by the two sensors, as illustrated in Figure 7. The critical region represents imminent danger to the patient and must be avoided at all times; the alarming region is not immediately dangerous but raises clinical concerns.

The control policy for the safety interlock may be to stop infusion as soon as the patient state enters the alarming region. The immediate challenge is to define the alarming region to be large enough so that the pump can always be stopped before the patient enters the critical region. At the same time, the region should not be too large to avoid false alarms which decrease the effectiveness of pain control unnecessarily. Finding the right balance and defining exact boundaries of the two regions was beyond the scope of the case study.



(a) Closed-loop PCA System.

(b) Regions of Patient's

Figure 7. Design of a PCA safety interlock.

The goal of the case study was to verify that the closed-loop system satisfies its patient requirements. To achieve this goal, one needs models of the infusion pump, pulse oximeter, control algorithm, and physiology of the patient.

Patient modeling is the critical aspect of the case study. Both pharmacokinetic and pharmacodynamics aspects of physiology should be considered [Mazoit07]. Pharmacokinetics specify how the internal state of the patient, represented by the drug concentration in the blood, is affected by the rate of infusion.

Pharmacodynamics specify how the internal state affects observable outputs of the model, that is, the relationship between the drug concentration and oxygen saturation levels measured by the pulse oximeter. The proof-of-concept approach taken in the case study relies on the simplified pharmacokinetic model of [Bequette03]. To make the model applicable to a diverse patient population, parameters of the model were taken to be ranges, rather than fixed values. To avoid the complexity of pharmacodynamics, a linear relationship between the drug concentration and patient vital signs is assumed.

Verification efforts concentrated on the timing of the control loop. After the patient enters the alarming region, it takes time for the controller to detect the danger and act on it. There are delays involved in obtaining sensor readings, delivering the readings from the pulse oximeter to the controller, calculating the control signal, delivering to the pump, and finally stopping the pump motor. In order to make verification results sound, the continuous dynamics of the patient model is used to derive t_{crit}, the minimum time over all combinations of parameter values in the patient model that can pass from the moment the patient state enters the alarming region to the moment it enters the critical region. The verification can now abstract away from the continuous dynamics, significantly simplifying the problem. Using a timing model of the components in the system, one can verify that the time it takes to stop the pump is always smaller than t_{crit}.

3.6.4 Remarks

The PCA system is a relatively simple but useful use case of closed-loop medical devices. It is worth noting that other types of closed-loop systems may introduce new engineering challenges due to their functionalities and requirements. For example, blood glucose control for diabetes has garnered a lot of attention from both engineering and clinical communities, and various concepts of closed-loop or semi-closed-loop systems have been proposed [Cobelli09, Kovatchev09, Hovorka04]. Compared to the PCA system, the closed-loop glucose control system is substantially more complex and opens many opportunities for new research.

The fail-safe mode in the PCA system is closely related to the clinical objective: overdosing is the major concern; while the patient may suffer from more pain when PCA is stopped, it is considered a safe action, at least for a reasonably time duration. Such kind of fail-safe mode may not exist in other clinical scenarios: for example, in the glucose control system, the goal is to keep the glucose level within a target range, i.e., stopping the insulin pump is not a default safe action since high glucose level is also harmful.

The safety criteria in the PCA system is defined by delineating a region in the state space of the patient model (such as the critical region in the case study above). Safety violations are then detected as threshold crossings in the stream of patient vital signs. Such crisp, threshold-based rules are often crude simplifications. Physiological systems have certain level of resilience and the true relation between health risks and physiological variables is still not completely understood. Time of exposure is also important: a short spike in the drug concentration may be less harmful than a longer interval of a lower-level concentration.

The sensor, pulse oximeter, used in the PCA system is relatively accurate with respect to the ranges that clinicians would concern. In some other scenarios, sensor accuracy is a non-negligible factor. For example, a glucose sensor can have a relative error of up to 15% [Ginsberg09], and given that the target range is relatively narrow, such error may significantly impact system operation and must be explicitly considered in the safety arguments.

Even if the sensor is perfectly accurate, it may not be predictive enough. While oxygen saturation can be used to detect respiratory failure, the effects could appear too late, that is, after harm to the patient is already done. Capnography data, which measures levels of carbon dioxide exhaled by the patient, can be used to detect the problem much sooner, but it is more expensive and invasive technology compared to pulse oximetry. This example shows the need to include more accurate pharmacodynamics data into the patient model, which can be used to account for the detection delay.

Another important factor in the closed-loop medical system is the human user's behavior. In the PCA system, the user behavior is relatively simple: the clinicians will be alerted in certain conditions, and most of the times they do not need to intervene in the operation of the control loop. In other applications with more complicated requirements, the user may demand a more involving role in the control. For example, in the glucose control application, a user will need to take back the control authority when the glucose level is significantly out of range, or even when the automatic controller is running, the user may choose to disapprove certain control actions for various reasons (e.g., the patient is not comfortable with a large insulin dose). The more complicated user interaction pattern introduces new challenges to the model-based validation and verification.

3.7 Assurance Cases

The safety of medical systems is of great public concern which is reflected in the fact that many such systems much adhere to government regulations and/or be certified by licensing bodies [Isaksen97]. For example, medical devices sold in the United States are regulated by the U.S. Food and Drug Administration (FDA). Some of these medical devices, such as infusion pumps, cannot be commercially distributed before receiving an approval from the FDA. There is a need to communicate, review and debate the trustworthiness of systems with a range of stakeholders (e.g., medical device manufacturers, and regulatory authorities). Assurance cases can be used to justify the adequacy of medical device systems. The assurance case is a method for arguing that a "body" of evidence justifies a claim. An assurance case addressing safety is called a safety case. A safety assurance case presents an argument, supported by a body of evidence, that a system is acceptably safe to be used in a given context [Menon09]. The notion of safety cases is currently embraced by several European industry sectors (e.g., aircraft, train, nuclear). More recently in the United States, the FDA issued draft guidance for medical infusion pump manufacturers to provide a safety case with their pre-market submissions [FDA10]. Infusion pump manufacturers are expected not only to achieve safety but also to convince regulators that it has been achieved [Ye05] through the submitted safety case. The manufacturer's role is to develop and submit a safety case to regulators to show that their product is acceptably safe to operate in the intended context [Kelly98]. The regulator's role is to assess the submitted safety case and make sure that the system is really safe.

There are many different approaches to the organization and presentation of safety cases. Goal Structuring Notation (GSN) is one description technique that has proven useful for constructing safety cases [Kelly04]. GSN is a graphical argumentation notation developed at the University of York. A GSN diagram includes elements that represent goals, argument strategies, contexts, assumptions, justifications, and evidence. The principal purpose of any goal structure in GSN is to show how goals, claims about the system specified with text within rectangular elements, are supported by a valid and convincing argument. To this end, goals are successively decomposed into sub-goals through implicit or explicit strategies. Strategies, specified with text within parallelograms, explicitly define how goals are decomposed into sub-goals. The decomposition continues until a point is reached where claims are supported by direct reference to available evidence, and the solution specified with text within circles. Assumptions/justifications, which define the rationale of the decomposition approach, are represented with ellipses. The context in which goals are stated is given in rectangle with rounded sides.

Another popular description technique is called Claims, Arguments, Evidence (CAE) notation [Adelard13]. While its notation is less standardized than GSN, it shares the same element types as GSN. The primary difference is that strategy elements are replaced with argument elements. In this work, we use GSN notation in presenting safety cases.

3.7.1 Challenges

The objective of a safety case development process is to provide justifiable rationale for design and engineering decisions and to instill confidence in those design decisions (in the context of system behavior) with stakeholders (e.g., manufacturers, and regulatory authorities). Adopting assurance cases necessarily requires the existence of proper reviewing mechanisms. These form the main aspects of assurance cases, i.e., building, trusting, and reviewing assurance cases.

There are challenges attached to the three aspects of assurance cases. These challenges need to be addressed to make safety cases practically useful:

• *Building assurance cases*. There exists a widely used method for systematically constructing safety cases. This method is often referred to as the "Six-Step" method [Kelly98a]. Following the "Six-Step" or any other method does not prevent mistakes that are commonly made by safety case developers, e.g., leaps from claims to evidence. Capturing successful (convincing, sound, etc.) arguments used in safety cases and reusing them in constructing new safety cases would minimize mistakes that may be made during the safety case development. The need for argument reusability motivates the use of the pattern concept (pattern means model or original used as archetypes) in the safety case constructions. Predefined patterns provide an inspiration or a starting point for new safety case developments. Using patterns would help improving safety cases maturity and completeness. Consequently, patterns can help device manufacturers to construct safety cases in a more efficient way in terms of completeness and development period. The concept of safety case patterns is defined in [Kelly97] to provide a way of capturing and reusing "best practice" in safety cases. "Best practice" captures company expertise, successfully certified approaches, etc. For example, patterns extracted from a safety case built for a specific product can be reused in constructing safety cases for other products that are developed via similar processes. Many safety case patterns were introduced in [Alexander07, Kelly98, Weaver03, Hawkins09, Wagner10, Ayoub12] to capture best practices.

• *Trusting assurance cases*. Although creating a structured safety case explicitly explains how the available evidence supports the overall claim of acceptable safety, it cannot ensure that the argument itself is 'good' (i.e., sufficient for its purpose) or the evidence is sufficient. Safety arguments typically have some weaknesses and so it cannot be fully trusted on its own. In other words, there is always a question about the trust in safety arguments and cited evidence, and so a justification for the sufficiency of confidence in safety cases is essential. There are attempts to quantitatively measure of confidence in safety cases such as [Bloomfield07, Denney11]. A new approach for creating clear safety cases was introduced in [Hawkins11] to facilitate the development process for safety cases and increase confidence in the constructed cases. This approach basically separates the major components of safety cases into safety argument and confidence argument. A safety argument is limited to give arguments and evidence that directly target the system safety. For example, claiming why a specific hazard is sufficiently unlikely to occur and arguing this claim by testing results as evidence. A confidence argument is given separately to justify the sufficiency of confidence in this safety argument. For example, questioning about the confidence in the given testing result evidence (e.g., is that testing exhaustive?) should be addressed in the confidence argument. These two components are given explicitly and separately. They are interlinked so that justification for having sufficient confidence in individual aspects of the safety component is clear and readily available but not confused with the safety component itself.

Any gap that prohibits perfect confidence in safety arguments is referred to as an assurance deficit [Hawkins11]. Argument patterns for confidence arguments are given in [Hawkins11]. Those patterns are defined based on identifying and managing the assurance deficits to show sufficient confidence in the safety argument. To this end, it is necessary to identify the assurance deficits as completely as practicable. Following a systematic approach (such as the one proposed in [Ayoub12a]) would help in effectively identifying assurance deficits. In [Menon09, Weaver03], lists of major factors that should be considered in determining the confidence in arguments are defined. Questions to be considered when determining the sufficiency of each factor are also given. To show sufficient confidence in a safety argument, a confidence argument developer first explores all concerns about the confidence in this argument, and then makes claims that these concerns are addressed. If a claim cannot be supported by convincing evidence, then a deficit is identified. The list of the recognized assurance deficits can be then used in instantiating the confidence pattern given in [Hawkins11] to show that the residual deficits are acceptable.

- *Reviewing assurance cases*. Safety case arguments are rarely provable deductive arguments. Instead they are more commonly inductive. And so safety cases are, by their nature, often subjective [Kelly07]. The objective of safety case evaluation, therefore, is to assess if there is a mutual acceptance of the subjective position. The human mind does not deal well with complex inferences based on uncertain sources of knowledge [Cyra08], which is common in safety arguments. Therefore, reviewers should only be required to express their opinions about the basic elements in the safety case. Then, a mechanism should provide a way to aggregate the reviewer opinions about the basic elements in the safety case to communicate a message about the overall sufficiency of it.

There are several approaches that have proposed. The work in [Kelly07] present a structured approach to assurance case review by focusing primarily on helping to assess the level of assurance offered by the assurance case argument. The work in [Goodenough12] outlines a framework for justifying confidence in the truth of assurance case claims. The framework is based on the notion of eliminative induction– the principle that confidence in the truth of a claim increases as reasons of doubting its truth are identified and eliminated. Defeaters offer possible reasons for doubting. Then the notion of Baconian probability is used to provide a measure for confidence in assurance cases based on how many defeaters have been identified and eliminated. In [Ayoub13] a structured method for assessing the level of sufficiency and insufficiency of safety arguments was outlined. The reviewer assessments and the results of their aggregation are represented in the Dempster-Shafer model [Sentz02]. The assessing mechanism given in [Ayoub13] can be used in conjunction with the step-by-step review approach proposed in [Kelly07] to answer the question given in the last-step of this reviewing approach, which is about the overall sufficiency of the safety argument. In other words, the approach in [Kelly07] provides a skeleton for a systematic review process; however, the mechanism in [Ayoub13] provides a systematic procedure to measure the sufficiency and insufficiency of the safety arguments. An appraisal mechanism is proposed in [Cyra08] to assess the trust cases using the Dempster-Shaffer model. Additionally, linguistic scales are introduced in [Cyra08] to express the expert opinions and the aggregation results. Linguistic scales are appealing as they are closer to human nature than numbers. They are based on qualitative values such as "high," "low," and "very low" and are mapped into the interval for evaluation.

3.7.2 Case Study: The GPCA safety case

This section builds on the case study of the GPCA infusion pump, presented in Section 3.3.3. Assurance cases for medical devices have been discussed in [Weinstock09]. The work in [Weinstock09] can be used as staring point for the GPCA safety case construction. A safety case given in [Jee10] is constructed for a pacemaker that is developed following a model-based approach similar to the one used in the GPCA case study.

Safety case patterns.

Similarities in the development approach are likely to lead to similarities in the safety argument. Safety case patterns [Kelly97] have been proposed as means of capturing similarities between arguments. Patterns allow the common argument structure be elaborated with device-specific details. To capture the common argument structure for systems developed in a model-based fashion, a safety case pattern, called the *from_to* pattern, has been proposed in [Ayoub12]. Below, the *from_to* pattern is illustrated and instantiated for the GPCA reference implementation.

A safety case for the GPCA reference implementation would claim that the PCA implementation software does not contribute to the system hazards when used in the intended environment. To address this claim, one needs to show that the PCA implementation software satisfies the GPCA safety requirements in the intended environment. This is the starting point for the pattern (see claim G1 in Figure 9). The context for this claim is that GPCA safety requirements are defined to mitigate the GPCA hazards, which would be argued separately in another part of the safety case.

Figure 8 shows the GSN structure of the proposed *from_to* pattern. Here, {to} refers to the system implementation and {from} refers to a model of this system. The claim (G1) about the implementation correctness (i.e., satisfaction of some property (referenced in C1.3)) is justified not only by validation (G4 through S1.2) but also by arguing over the model correctness (G2 through S1.1), and the consistency between the model and the implementation created based on it (G3 through S1.1). The model correctness (i.e., further development for G2) is guaranteed through the model verification (i.e., the second step of the model-based approach). The consistency between the model and the implementation (i.e., further development for G3) is supported by the code generation from the verified model (i.e., the third step of the model-based approach). Only part of the property of concern (referenced in C2.1) can be verified at the model level due to the different abstraction levels between the model and the implementation. However, the validation argument (S1.2) covers the entire property of concern (referenced in C1.3). The additional justification given in (S1.1) increases the assurance in the top-level claim (G1).

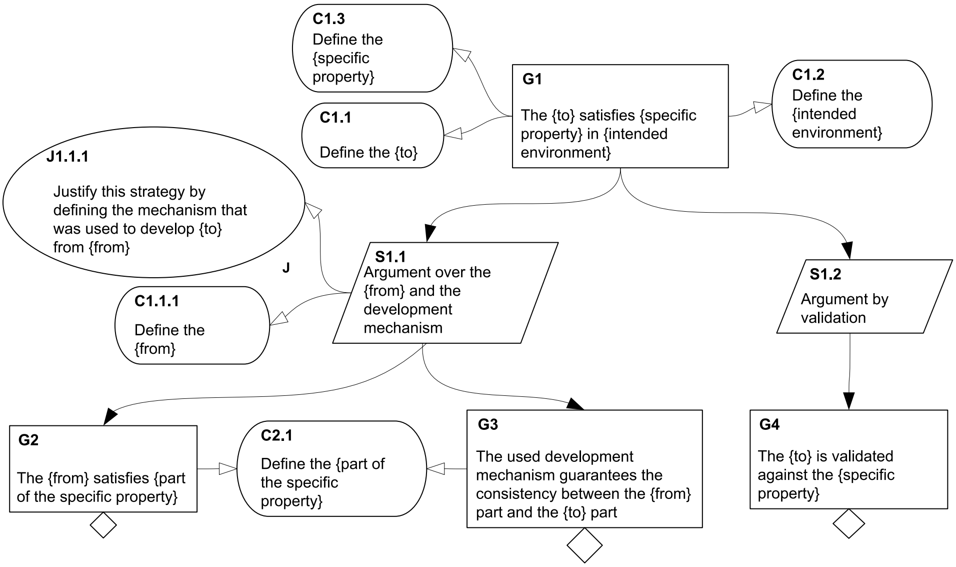

C1.3
Define the {specific property}
C1.1
Define the {to}
G1
The {to} satisfies {specific property} in {intended environment}
C1.2
Define the {intended environment}
J1.1.1
Justify this strategy by defining the mechanism that was used to develop {to} from {from}
J
C1.1.1
Define the {from}
S1.1
Argument over the {from} and the development mechanism
S1.2
Argument by validation
G2
The {from} satisfies {part of the specific property}
C2.1
Define the {part of the specific property}
G3
The used development mechanism guarantees the consistency between the {from} part and the {to} part
G4
The {to} is validated against the {specific property}

Figure 8: The proposed *from_to* pattern

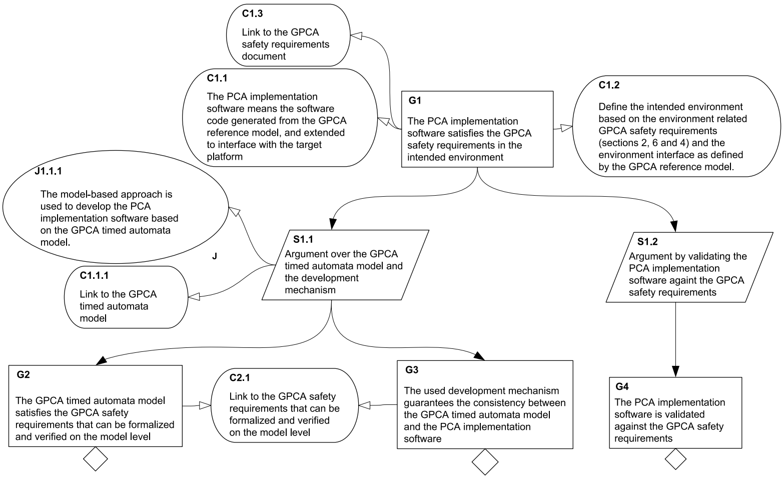

C1.3
Link to the GPCA safety requirements document
C1.1
The PCA implementation software means the software code generated from the GPCA reference model, and extended to interface with the target platform
G1
The PCA implementation software satisfies the GPCA safety requirements in the intended environment
C1.2
Define the intended environment based on the environment related GPCA safety requirements (sections 2, 6 and 4) and the environment interface as defined by the GPCA reference model.
J1.1.1
The model-based approach is used to develop the PCA implementation software based on the GPCA timed automata model.
J
S1.1
Argument over the GPCA timed automata model and the development mechanism
S1.2
Argument by validating the PCA implementation software againt the GPCA safety requirements
C1.1.1
Link to the GPCA timed automata model
G2
The GPCA timed automata model satisfies the GPCA safety requirements that can be formalized and verified on the model level
C2.1
Link to the GPCA safety requirements that can be formalized and verified on the model level
G3
The used development mechanism guarantees the consistency between the GPCA timed automata model and the PCA implementation software
G4
The PCA implementation software is validated against the GPCA safety requirements

Figure 9: An instance of the *from_to* pattern

Figure 9 shows an instantiation of this pattern that is part of the PCA safety case. Based on [Kim11], for this pattern instance, the {to} part is the PCA implementation software (referenced in C1.1), the {from} part is the GPCA timed automata model (referenced in C1.1.1) and the GPCA safety requirements (referenced in C1.3) represent the concerned property. In this case, correct PCA implementation means it satisfies the GPCA safety requirements that defined to guarantee the PCA safety. The satisfaction of the GPCA safety requirements in the implementation level (G1) is decomposed by two strategies (S1.1) and (S1.2). The argument in (S1.1) is supported by the correctness of the GPCA timed automata model (G2), and the consistency between the model and the implementation (G3). The correctness of the GPCA timed automata model (i.e., further development for G2) has been proved using the UPPAAL model-checker against the GPCA safety requirements that can be formalized (referenced in C2.1). The consistency between the model and the implementation (i.e., further development for G3) is supported by the code-synthesis from the verified GPCA timed automata model. Not all the GPCA safety requirements (referenced in C1.3) can be verified against the GPCA timed automata model [Kim11]. Only the part referenced in C2.1 can be formalized and verified in the model level (e.g., "*no bolus dose shall be possible during the Power-On Self-Test*"). Other requirements cannot be formalized and/or verified against the model given its level of details (e.g., "*the flow rate for the bolus dose shall be programmable*" cannot be formalized meaningfully and then verified in the model level).

Note.

Generally, using safety case patterns does not necessarily guarantee that the constructed safety case will be sufficiently compelling. So when instantiating the *from_to* pattern, it is necessary to be able to provide justification for each instantiation decision to guarantee that the constructed safety case is sufficiently compelling. Assurance deficits should be identified throughout the construction of a safety argument. Where an assurance deficit is identified, it is necessary to demonstrate that the deficit is either acceptable, or addressed such that it becomes acceptable. An explicit justification should be provided as to why the residual assurance deficit is considered acceptable. This can be done by adopting appropriate approaches such as ACARP (As Confidence As Reasonably Practical) [Hawkins09a].

Assurance deficit example.

As discussed in Section 3.3.3 and showed in Figure 3, the GPCA Simulink/Stateflow model was transformed into an equivalent GPCA timed automata model. Although it is relatively straight forward to translate the original GPCA model written in Simulink/Stateflow into a UPPAAL timed automata model, there is no explicit evidence to show the equivalence between the two models at the semantic level. A potential assurance deficit associated with the GPCA timed automata model (context C1.1.1, Figure 9) can be formed as "there are semantic differences between the Simulink/Stateflow and the UPPAAL timed automata model." For this residual assurance deficit, exhaustive conformance testing between the GPCA Simulink/Stateflow model and the GPCA timed automata model may be a reasonable mitigation.

3.7.3 Remarks

Recently, safety cases have become popular and acceptable ways for communicating ideas and information about the safety-critical systems among the system stakeholders. In the medical device domain, the FDA issued draft guidance for medical infusion pump manufacturers to provide a safety case with their pre-market submissions [FDA10]. In this section, a brief introduction about safety cases and notations used to describe them are discussed. Three aspects of safety cases to make them practically useful are listed and discussed, which are facilitating safety cases construction, justifying the existence of sufficient trust in safety arguments and cited evidence, and providing a framework for safety case assessment for regulation and certification.

Safety case patterns can help both device manufacturers and regulators to construct and review the safety cases more efficiently while improving confidence and shortening an approval period. The qualitative reasoning about the confidence existence is believed to be more consistent with the inherited subjectivity in safety cases than the quantitatively reasoning. The separation between safety and confidence arguments reduces the size of the core safety argument. Consequently, this structure is believed to facilitate the development and reviewing processes for safety cases. The constructed confidence arguments should be used in the appraisal process for assurance arguments as illustrated in [Cyra08, Kelly07, Ayoub13].

Given the subjectivity nature of safety case, the review methods cannot replace the reviewer; instead they form frameworks to lead safety case reviewers through the evaluation process. Consequently, the result of the safety case review process is always subjective.

4 Practitioner's Implications

One can distinguish the following groups of stakeholders in MCPS: (1) MCPS developers, including manufacturers of medical devices and integrators of medical information technologies; (2) MCPS administrators, typically clinical engineers in hospitals, who are tasked with deploying and maintaining MCPS; (3) MCPS users, clinicians who perform treatment using MCPS; (4) MCPS subjects, that is, patients; and (5) MCPS regulators, who hold the mandate for certifying the safety of MCPS or approving their use for clinical purposes. In the United States, the Food and Drug Administration is the regulatory agency charged with assessing safety and effectiveness of medical devices and their approval for specific uses. All of these groups have a vested interest in MCPS safety. However, each group has additional drivers that need to be taken into account when designing or deploying MCPS in a clinical setting. Below, we consider each group of stakeholders and identify specific concerns that apply to them, and additional challenges they pose.

4.1 MCPS Developer Perspective

Dependence of MCPS on software, as well as complexity of software used in medical devices, has been steadily increasing over the past three decades. In recent years, medical device industry has been plagued with software-related recalls, with 19% of all recalls of medical devices in the U.S. being related to software problems [Simone13].

Many other safety-regulated industries, such as avionics and nuclear power, operate on relatively long design cycles. By contrast, medical device companies are under heavy market pressure to quickly introduce additional features into their products. At the same time, medical devices are often developed by relatively small companies that lack the resources for extensive validation and verification of each new feature they introduce. Model-based development techniques, such as the ones described in Section 3.3, hold the promise of more efficient verification and validation, leading to shorter development cycles.

At the same time, many companies complain about heavy regulatory burden imposed by the FDA and similar regulatory agencies in other countries. Formal models and verification results, introduced by the model-based development approaches, provide evidence that MCPS is safe. Combined with the assurance cases that organize this evidence into a safety argument, these rigorous development methods may help reduce the regulatory burden for MCPS developers.

4.2 MCPS Administrator Perspective

Clinical engineers in hospitals are charged with maintaining a wide variety of medical devices that comprise the MCPS used in patient treatment. Most clinical scenarios today involve multiple medical devices. A clinical engineer needs to ensure that the devices used in treating a patient can all work together. If an incompatibility is discovered after treatment commences, the patient may be harmed. Interoperability techniques, described in Section 3.4, may help to ensure that more devices are compatible with each other, making the job of maintaining the inventory and assembly of clinical scenarios easier. This, in turn, reduces treatment errors and improves patient outcomes and, at the same time, saves hospital money.

4.3 MCPS User Perspective

Clinicians use MCPS to perform patient treatment. A specific treatment can, in most cases, be performed using different MCPS implementations using similar devices from different vendors. A primary concern, then, is ensuring that clinicians are equally familiar with the different implementations. The concepts of a clinical scenarios and virtual medical devices, introduced in Section 3.4 can help establish a common user interface for the MCPS, regardless of which devices are used to implement it. Such an interface would help to reduce clinical errors in using the devices. Furthermore, the user interface can be verified as part of the analysis of the MCPS model, as suggested by [Masci13].

MCPS development must take existing standards of care into consideration. Clinical personnel needs to be involved in the analysis of the scenario models to ensure that they are consistent with extant clinical guidelines for the respective treatment and are intuitive for caregivers to use.

A particular challenge in modern healthcare is the high workload faced by caregivers. Each caregiver is caring for multiple patients and has to keep track of multiple sources of information about each patient. On-demand MCPS have the potential to control cognitive overload in caregivers by offering virtual devices that offer intelligent presentation of clinical information or smart alarm functionality.

Smart alarms, which can correlate or prioritize alarms from individual devices, can be of great help to caregivers, by giving a more accurate picture of patient state and reducing the rate of false alarms [Imhoff09].

4.4 Patient Perspective

Arguably, of all stakeholder groups, patients stand to gain the most from the introduction of MCPS. In addition to the expected improvements in the safety of treatments through higher reliability of individual devices and their bedside assemblies, patients would get the benefit of improvements in treatments themselves. These improvements may come from several sources.

On the one hand, MCPS can offer continuous monitoring that caregivers, who normally attend multiple patients, cannot provide by themselves. Clinical guidelines often require caregivers to obtain patient data at fixed intervals; for example, every 15 minutes. An MCPS may collect patient data as frequently as allowed by each sensor and alert caregivers to changes in the patient's condition earlier and let them interfere before the change leads to a serious problem. Furthermore, continuous monitoring, combined with support for predictive decision making, similar to the one discussed in Section 3.5, will allow treatment to be proactive rather than reactive.

Probably the biggest improvement in the quality of care for patients may come with the transition from general guidelines meant to apply to all patients within a certain population to personalized approaches, where treatment is customized to individual needs of the patient and takes into account personalized characteristics. Personalized treatments, however, cannot be effected without detailed patient models. Such models can be stored in patient records and interpreted by the MCPS during treatment.

4.5 MCPS Regulatory Perspective

Regulators of medical device industry are tasked with assessing safety and effectiveness of MCPS. The two main concerns that the regulators face are improving the quality of the assessment and making the best use of limited resources that agencies have for performing the assessment. These two concerns are not independent, because more efficient ways of performing assessments allow regulators more time to explore deeper in their evaluation. Safety case technologies discussed in Section 3.7 may help address both. The move towards evidence-based assessment may allow regulators to perform more accurate and reliable assessment. At the same time, organizing evidence into a coherent argument helps to perform the assessment more efficiently.

5 Summary and Open Challenges

We presented a broad overview of trends in MCPS and design challenges that these trends present. We also discussed possible approaches to address these challenges, based on recent results in MCPS research.

The first challenge is related to the prevalence of software-enabled functionality in modern MCPS, which makes assurance of patient safety a much harder task. Model-based development techniques provide one way to ensure safety of the system. Increasingly, model-based development is embraced by medical device industry. Still, numerous recent recalls demonstrate that the problem of device safety is far from being solved.

The next-level challenge arises from the need to compose individual device into a system of interconnected devices that collectively treat the patient in a complex clinical scenario. Such multi-device MCPS can provide new modes of treatment, give enhanced feedback to the clinician, and improve patient safety. However, additional hazards can arise from communication failures and lack of interoperability between devices. Reasoning about safety of such on-demand MCPS, which are assembled at bedside from available devices, creates new regulatory challenges and requires medical application platforms, trusted middleware that will ensure correct interactions between the devices. Research prototypes of such middleware are currently being developed, but their effectiveness needs to be further evaluated. Furthermore, interoperability standards for on-demand MCPS need to be further improved and gain wider acceptance.

In order to fully utilize the promise of multi-device MCPS, new algorithms need be developed to process and fuse patient data from multiple sensors, provide better decision support for clinicians, more accurate and informative alarms, etc. This need gives rise to two kinds of open challenges. On the one hand, additional clinical as well as data analysis research needs to be performed, to determine the best ways to utilize the new information made available through combining multiple rich data sources. On the other hand, there is a need for new software tools to facilitate fast prototyping and deployment of new decision support and visualization algorithms.

MCPS promises to enable a wide array of physiological closed-loop systems, where the information about the patient state, collected from multiple sensors, can be used to adjust the treatment process or its parameters. Research on such closed-loop control algorithms is gaining prominence, especially for glycemic control for diabetes patients. However, much research needs to be performed to better understand patient physiology and develop adaptive control algorithms that can deliver personalized treatment to each patient.

In all of these applications, patient safety and effectiveness of treatment are the two paramount concerns. MCPS manufacturers need to convince regulators that systems they build are safe and effective. The growing complexity of MCPS, high connectivity, and prevalence of software-enabled functionality make evaluation of system safety quite difficult. Construction of effective assurance cases for MCPS, as well as for CPS in general, remains a challenge in need of further research.

References

[Adelard13]. Adelard. Claims, Arguments and Evidence (CAE). http://plato.stanford.edu/entries/reasoning-defeasible/, 2013.

[Alexander07]. R. Alexander, T. Kelly, Z. Kurd, and J. Mcdermid. Safety Cases for Advanced Control Software: Safety Case Patterns. Technical report, University of York, 2007.

[Amnell03]. 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 FORMATS, 2003.

[Arney09]. D. Arney, J. M. Goldman, S. F. Whitehead, and I. Lee. Synchronizing an x-ray and anesthesia machine ventilator: A medical device interoperability case study. In BIODEVICES 2009, pages 52 – 60, January 2009.

[ASTM09]. ASTM International. ASTM F2761-2009. 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.

[Ayoub13]. A. Ayoub, J. Chang, O. Sokolsky, and I. Lee. Assessing the Overall Sufficiency of Safety Arguments. In Safety Critical System Symposium (SSS'13), 2013.

[Ayoub12]. A. Ayoub, B. Kim, I. Lee, and O. Sokolsky. A Safety Case Pattern for Model-Based Development Approach. In NFM2012, pages 223–243, Virginia, USA, 2012.

[Ayoub12a]. A. Ayoub, B. Kim, I. Lee, and O. Sokolsky. A Systematic Approach to Justifying Sufficient Confidence in Software Safety Arguments. In International Conference on Computer Safety, Reliability and Security (SAFECOMP 2012), Magdeburg, Germany, 2012.

[Becker09]. U. Becker. Model-based development of medical devices. In Proceedings of the Workshop on Computer Safety, Reliability, and Security (SAFECERT '09), volume 5775 of LNCS, pages 4–17, 2009.

[Behrmann04] .G. Behrmann, A. David, and K. Larsen. A tutorial on UPPAAL. In Formal Methods for the Design of Real-Time Systems, LNCS, pages 200–237, 2004.

[Bequette03].B. Bequette. Process control: modeling, design, and simulation. Prentice Hall Press, 2003.

[Bloomfield07].R. Bloomfield, B. Littlewood, and D. Wright. Confidence: Its Role in Dependability Cases for Risk Assessment. In Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on, pages 338 –346, 2007.

[Carr03].C. D. Carr and S. M. Moore. Ihe: a model for driving adoption of standards. Computerized Medical Imaging and Graphics, 27(2â€"3):137 – 146, 2003. <ce:title>Picture Archiving and Communication Systems 20 Years Later</ce:title>.

[Clarke07]. M. Clarke, D. Bogia, K. Hassing, L. Steubesand, T. Chan, and D. Ayyagari. Developing a standard for personal health devices based on 11073. In Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pages 6174–6176, 2007.

[Clifford09].G. Clifford, W. Long, G. Moody, and P. Szolovits. Robust parameter extraction for decision support using multimodal intensive care data. Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences, 367:411–429, 2009.

[Clinical07].Clinical Alarms Task Force. Impact of clinical alarms on patient safety. Journal of Clinical Engineering, 32(1):22–33, 2007.

[Cobelli09].C. Cobelli, C. D. Man, G. Sparacino, L. Magni, G. D. Nicolao, , and B. P. Kovatchev. Diabetes: Models, signals, and control. Biomedical Engineering, IEEE Reviews in, 2, 2009.

[Commission13].J. Commission. Sentinel event alert issue 50: Medical device alarm safety in hospitals. 50, April 2013.

[Cyra08].L. Cyra and J. Górski. Expert Assessment of Arguments: A Method and Its Experimental Evaluation. In SAFECOMP, 2008.

[Denney11].E. Denney, G. Pai, and I. Habli. Towards Measurement of Confidence in Safety Cases. In International Symposium on Empirical Software Engineering and Measurement (ESEM'11), Washington, DC, USA, 2011. IEEE Computer Society.

[Dias07].A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos. A survey on model-based testing approaches: a systematic review. In Proceedings of the 1 ACM international workshop on Empirical assessment of software engineering languages and technologies, pages 31–36, 2007.

[Dolin06].R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, P. V. Biron, and A. S. Shvo. Hl7 clinical document architecture, release 2. Journal of the American Medical Informatics Association, 13(1):30–39, 2006.

[Donchin02].Y. Donchin and F. J. Seagull. The hostile environment of the intensive care unit. Current Opinion in Critical Care, 8:316–320, 2002.

[Edworthy06].J. Edworthy and E. Hellier. Alarms and human behaviour: implications for medical alarms. British Journal of Anaesthesia, 97:12–17, 2006.

[EBMWG92].Evidence-Based Medicine Working Group. Evidence-based medicine: A new approach to teaching the practice of medicine. Journal of the American Medical Association, 268:2420–2425, 1992.

[Harris13].Harris healthcare (formerly careFX). http://healthcare.harris.com/solutions/default.aspx, 2013.

[Garg05].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. Journal of the American Medical Association, 293:1223–1238, 2005.

[Ginsberg09].B. H. Ginsberg. Factors affecting blood glucose monitoring: Sources of errors in measurement. Journal of Diabetes Science and Technology, 3(4):903–913, 2009.

[Goldman05].J. Goldman, R. Schrenker, J. Jackson, and S. Whitehead. Plug-and-play in the operating room of the future. Biomedical Instrumentation and Technology, 39(3):194–199, 2005.

[Goodenough12]. J. Goodenough, C. Weinstock, and A. Klein. Toward a Theory of Assurance Case Confidence. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Technical Report CMU/SEI-2012-TR-002, 2012.

[Hatcliff12]. J. Hatcliff, A. King, I. Lee, A. Macdonald, A. Fernando, M. Robkin, E. Vasserman, S. Weininger, and J. M. Goldman. Rationale and architecture principles for medical application platforms. In Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems, ICCPS '12, pages 3–12, Washington, DC, USA, 2012. IEEE Computer Society.

[Hawkins11].R. Hawkins, T. Kelly, J. Knight, and P. Graydon. A New Approach to creating Clear Safety Arguments. In SSS'11, pages 3–23. Springer London, 2011.

[Hawkins09].R. Hawkins and T. Kelly. A Systematic Approach for Developing Software Safety Arguments. Journal of System Safety, 46:25–33, 2009.

[Hawkins09a].R. Hawkins and T. Kelly. Software Safety Assurance – What Is Sufficient? In 4th IET International Conference of System Safety, 2009.

[Henzinger07].T. A. Henzinger and C. M. Kirsch. The embedded machine: Predictable, portable real-time code. ACM Transactions on Programming Languages and Systems (TOPLAS), 29(6):33, 2007.

[Hovorka04].R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. 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. Physiological Measurement, 25(4):905, 2004.

[Imhoff09]. M. Imhoff, S. Kuhls, U. Gather, and R. Fried. Smart alarms from medical devices in the OR and ICU. Best Practice and Research in Clinical Anaesthesiology, 23(1):39–50, 2009.

[Imhoff06].M. Imhoff and S. Kuhls. Alarm algorithms in critical care monitoring. Anesthesia and Analgesia, 102(5):1525–1536, 2006.

[Isaksen97].U. Isaksen, J. P. Bowen, and N. Nissanke. System and Software Safety in Critical Systems. Technical Report RUCS/97/TR/062/A, The University of Reading, UK, 1997.

[Iso/ieee11073].Iso/ieee 11073 committee. http://standards.ieee.org/findstds/standard/11073-10103-2012.html.

[Jackson07].D. Jackson, M. Thomas, and L. I. Millett, editors. Software for Dependable Systems: Sufficient Evidence? National Academies Press, May 2007. Committee on Certifiably Dependable Software Systems, National Research Council.

[Jee10].E. Jee, I. Lee, and O. Sokolsky. Assurance cases in model-driven development of the pacemaker software. In 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part II, ISoLA'10, pages 343–356, Berlin, Heidelberg, 2010. Springer-Verlag.

[Jeroeno4]. J. C. H. JEROEN V. LEVERT. Runaway pacemaker due to software-based programming error. Pacing and Clinical Electrophysiology, 27(12):1689–1690, Dec. 2004.

[Kelly97].T. Kelly and J. McDermid. Safety Case Construction and Reuse using Patterns. In SAFECOMP, pages 55–96. Springer-Verlag, 1997.

[Kelly04].T. Kelly and R. Weaver. The goal structuring notation – a safety argument notation. In DSN 2004 Workshop on Assurance Cases, 2004.

[Kelly98].T. Kelly. Arguing safety – a systematic approach to managing safety cases. PhD thesis, Department of Computer Science, University of York, 1998.

[Kelly98a].T. Kelly. A six-step Method for Developing Arguments in the Goal Structuring Notation (GSN). Technical report, York Software Engineering, UK, 1998.

[Kelly07].T. Kelly. Reviewing Assurance Arguments – A Step-by-Step Approach. In Workshop on Assurance Cases for Security - The Metrics Challenge, Dependable Systems and Networks (DSN), 2007.

[Kim12]. B. G. Kim, L. T. Phan, I. Lee, and O. Sokolsky. A model-based i/o interface synthesis framework for the cross-platform software modeling. In Rapid System Prototyping (RSP), 2012 23rd IEEE International Symposium on, pages 16–22. IEEE, 2012.

[Kim11].B. Kim, A. Ayoub, O. Sokolsky, P. Jones, Y. Zhang, R. Jetley, and I. Lee. Safety-Assured Development of the GPCA Infusion Pump Software. In EMSOFT, pages 155–164, Taipei, Taiwan, 2011.

[King09].A. King, S. Procter, D. Andresen, J. Hatcliff, S. Warren, W. Spees, R. Jetley, P. Jones, and S. Weininger. An open test bed for medical device integration and coordination. In Proceedings of the 31st International Conference on Software Engineering, 2009.

[Kovatchev09].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, 3(1):44–55, 2009.

[Lee06].I. Lee, G. J. Pappas, R. Cleaveland, J. Hatcliff, B. H. Krogh, P. Lee, H. Rubin, and L. Sha. High-confidence medical device software and systems. Computer, 39(4):33–38, April 2006.

[Lee12].I. Lee, O. Sokolsky, S. Chen, J. Hatcliff, E. Jee, B. Kim, A. King, M. Mullen-Fortino, S. Park, A. Roederer, and K. Venkatasubramanian. Challenges and research directions in medical cyber-physical systems. Proceedings of the IEEE, 100(1):75–90, Jan 2012.

[Lofsky04].A. S. Lofsky. Turn Your Alarms On. APSF Newsletter, 19(4):43, 2004.

[Lublinerman09].R. Lublinerman, C. Szegedy, and S. Tripakis. Modular code generation from synchronous block diagrams: modularity vs. code size. In Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages, POPL '09, pages 78–89, New York, NY, USA, 2009. ACM.

[Lynn11].L. A. Lynn and J. P. Curry. Patterns of unexpected in-hospital deaths: a root cause analysis. Patient Safety in Surgery, 5, 2011.

[Maddox08].R. Maddox, H. Oglesby, C. Williams, M. Fields, and S. Danello. Continuous respiratory monitoring and a "smart" infusion system improve safety of patient-controlled analgesia in the postoperative period. In K. Henriksen, J. Battles, M. Keyes, and M. Grady, editors, Advances in Patient Safety: New Directions and Alternative Approaches, volume 4 of Advances in Patient Safety. Agency for Healthcare Research and Quality, Aug. 2008.

[Masci13].P. Masci, A. Ayoub, P. Curzon, I. Lee, O. Sokolsky, and H. Thimbleby. Model-based development of the generic PCA infusion pump user interface within PVS. In Proceedings of the 32 International Conference on Computer Safety, Reliability and Security (SAFECOMP '13), 2013. To appear.

[Mazoit07].J. X. Mazoit, K. Butscher, and K. Samii. Morphine in postoperative patients: Pharmacokinetics and pharmacodynamics of metabolites. Anesthesia and Analgesia, 105(1):70–78, 2007.

[MDCF].Medical Device Coordination Framework (MDCF) website. http://mdcf.santos.cis.ksu.edu.

[MDPNP].MD PnP: Medical Device "Plug-and-Play" Interoperability Program website. http://www.mdpnp.org.

[Menon09].C. Menon, R. Hawkins, and J. McDermid. Defence standard 00-56 issue 4: Towards evidence-based safety standards. In Safety-Critical Systems: Problems, Process and Practice, pages 223–243. Springer London, 2009.

[Nuckols08].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. Journal of General Internal Medicine, 23(Supplement 1):41–45, January 2008.

[Oberli99].C. Oberli, C. Saez, A. Cipriano, G. Lema, and C. Sacco. An expert system for monitor alarm integration. Journal of Clinical Monitoring and Computing, 15:29–35, 1999.

[Pajic12]. 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, PP(99):1–1, 2012.

[Phillips10].Phillips eICU program. http://www.healthcare.philips.com/main/products/patient_monitoring/products/eicu/index.wpd, 2010.

[Rae03].A. Rae, P. Ramanan, D. Jackson, and J. Flanz. Critical feature analysis of a radiotherapy machine. In International Conference of Computer Safety, Reliability and Security (SAFECOMP), Sept. 2003.

[Sapirstein09].A. Sapirstein, N. Lone, A. Latif, J. Fackler, and P. J. Pronovost. Tele ICU: paradox or panacea? Best Practice & Research Clinical Anaesthesiology, 23(1):115–126, Mar. 2009.

[Sentz02].K. Sentz and S. Ferson. Combination of evidence in Dempster-Shafer theory. Technical report, Sandia National Laboratories, SAND 2002-0835, 2002.

[Shortliffe79].E. H. Shortliffe, B. G. Buchanan, and E. A. Feigenbaum. Knowledge engineering for medical decision making: A review of computer-based clinical decision aids. Proceedings of the IEEE, 67:1207–1224, 1979.

[Simone13].L. K. Simone. Software related recalls: A forensic analysis of recall records. Biomedical Instrumentation & Technology, 2013. In press.

[McMaster13].Software Quality Research Laboratory, McMaster Univeristy. Pacemaker formal methods challenge. http://sqrl.mcmaster.ca/pacemaker.htm, accessed August 10, 2013.

[FDA10].U.S. Food and Drug Administration, Center for Devices and Radiological Health. Guidance for Industry and FDA Staff - Total Product Life Cycle: Infusion Pump - Premarket Notification [510(k)] Submissions, Apr. 2010.

[FDA10a].U.S. Food and Drug Administration, Center for Devices and Radiological Health. White Paper: Infusion Pump Improvement Initiative, Apr. 2010.

[Wagner10].S. Wagner, B. SchÃ¤tz, S. Puchner, and P. Kock. A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models. In ISSRE, pages 269–278, 2010.

[Weaver03].R. Weaver. The Safety of Software - Constructing and Assuring Arguments. PhD thesis, Department of Computer Science, University of York, 2003.

[Weinstock09].C. Weinstock and J. Goodenough. Towards an Assurance Case Practice for Medical Device. Technical report, CMU/SEI-2009-TN-018, 2009.

[Ye05].F. Ye. Contract-based justification for COTS component within safety-critical applications. PhD thesis, Department of Computer Science, University of York, 2005.

[UPenn].The generic patient controlled analgesia pump model. http://rtg.cis.upenn.edu/gip.php3.

[UPenn-a].Safety Requirements for the Generic Patient Controlled Analgesia Pump. http://rtg.cis.upenn.edu/gip.php3.

[UPenn-b].The Generic Patient Controlled Analgesia Pump Hazard Analysis. http://rtg.cis.upenn.edu/gip.php3.