SEI SERIES IN SOFTWARE ENGINEERING

Cyber-Physical Systems

Raj Rajkumar

Dionisio de Niz

Mark Klein

Cyber-Physical Systems

Cyber-Physical Systems

Raj Rajkumar
Dionisio de Niz
Mark Klein

Addison-Wesley

Boston • Columbus • Indianapolis • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Software Engineering Institute | Carnegie Mellon

The SEI Series in Software Engineering

Library of Congress Control Number: 2016953412

ISBN-13: 978-0-321-92696-8
ISBN-10: 0-321-92696-X

1 16

To our families

This page intentionally left blank

Contents

Introduction

Ragunathan (Raj) Rajkumar, Dionisio de Niz, and Mark Klein

The National Science Foundation defines cyber-physical systems (CPS) as "engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components"—that is, cyber and physical components. In practical terms, this integration means that, to understand CPS behavior, we cannot focus only on the cyber part or only on the physical part. Instead, we need to consider both parts working together. For instance, if we try to verify the behavior of the airbag of a car, it is not enough to ensure that the correct instructions to inflate the airbag are executed when the system detects that a crash is occurring. We also need to verify that the execution of such instructions is completed in sync with the physical process—for example, within 20 ms—so as to ensure that the airbag is fully inflated before the driver hits the steering wheel. The seamless integration between the cyber and physical parts of a CPS involves an understanding across multiple aspects that, in this simple example, include software logic, software execution timing, and physical forces.

While the airbag example already contains important elements of a CPS, we must note that it is one of the simplest examples and does not involve the most significant challenges for CPS. In particular, both the cyber components and the physical components are very simple in this case, and their interaction can be reduced to worst-case differences between the software execution completion time and the time it takes the driver to hit the steering wheel in a crash. However, as the complexity of both the software and the physical processes grows, the complexity of their integration also increases significantly. In a large CPS—for example, one of the latest commercial aircraft—the integration of multiple physical and cyber components and the tradeoffs between their multiple aspects become very challenging. For instance, the simple use of additional lithium ion batteries in the Boeing 787 Dreamliner brings a combination of multiple constraints that must be satisfied. In particular, we not only need to be able to satisfy the power consumption requirements under different operating modes with a specific configuration of batteries (interacting with software that is run at a specific processor speed and voltage), but we also need to manage when and how battery cells are charged and discharge while preserving the required voltage and verify that the charge/discharge configurations do not overheat the battery (causing it to burst into flames, as occurred during the first few flights of the 787), which interacts with the thermal dissipation design of the system. More importantly, all of these aspects, and more, must be certified under the strict safety standards from the Federal Aviation Administration (FAA).

CPS, however, are facing even more challenges than the increased complexity coming from single systems. Notably, they are being developed to interact with other CPS without human intervention. This phenomenon resembles the way the Internet began. Specifically, the Internet started as a simple connection between two computers. However, the real revolution happened when computers all over the world were connected seamlessly and a large number of services were developed on top of this worldwide network. Such connectivity not only allowed services to be delivered all over the globe, but also enabled the collection and processing of massive amounts of information in what is now known as "Big Data." Big Data allow us to explore trends among groups of people, or even explore trends in real time when they are combined with social media such as Facebook and Twitter. In CPS, this revolution is just starting. We can already observe services that collect driving information through global positioning system (GPS) applications on smartphones and allow us to select routes with low traffic congestion, thereby taking us in the direction of a smart highway, even if it is still mediated by humans. More significant steps in this direction are occurring every day, such as in the multiple projects involving autonomous cars. Most of these cars not only know how to drive a route autonomously, but also correctly interact with other non-autonomous cars on the route.

Emergence of CPS

Before CPS emerged as a specific, yet evolving discipline, systems composed of physical and cyber components already existed. However, the interactions between these two types of components were very simple, and the supporting theoretical foundations were mostly partitioned into silos. Specifically, the theoretical foundations for computer science and physical sciences where developed independently and ignored each other. As a result, techniques to, say, verify properties in one discipline such as thermal resilience, aerodynamics, and mechanical stress were developed independently from advances in computer science such as logical clocks, model checking, type systems, and so on. These advances, in fact, abstracted away behaviors that could be important in one discipline but were not relevant in the other. This is the case, for instance, with the timeless nature of programming languages and logical verification models in which only sequences of instructions are considered. This nature contrasts with the importance of time in the evolution of physical processes such as the movement of a car and maintenance of the temperature of a room that we are trying to control.

The early realization of the interactions between computational and physical science gave birth to some simple and mostly pairwise interaction models. This is the case with real-time scheduling theory and control theory, for example. On the one hand, scheduling theory adds time to computational elements and allows us to verify the response time of computations that interact with physical processes, thereby ensuring that such a process does not deviate beyond what the computational part expects and is capable of correcting. On the other hand, control theory allows us to put together a control algorithm and a physical process and analyze whether it would be possible for the algorithm to keep the system within a desired region around a specific setpoint. While control theory uses a continuous time model in which computations happen instantaneously, it allows the addition of delays to take into account the computation time (including scheduling), making it possible to specify the periodicity of computations and provide a clean interface with scheduling.

As the complexity of the interactions between domains increases, new techniques are developed to model such interactions. This is the case, for instance, with hybrid systems—a type of state machine in which the states model computational and physical states and the transitions model computational actions and physical evolutions. While these techniques enhance the capacity to describe complex interactions, their analysis is, more often than not, intractable. In general, the complexity of these models prevents the analysis of systems of practical dimensions. As the number of scientific disciplines that need to be considered grows (e.g., functional, thermal, aerodynamics, mechanical, fault tolerance), their interactions need to be analyzed to ensure that the assumptions of each discipline and its models are not invalidated by the other disciplines' models. For instance, the speed of a processor assumed by a real-time scheduling algorithm may be invalidated if the dynamic thermal management (DTM) system of the processor decreases the speed of this processor to prevent it from overheating.

CPS Drivers

While CPS are already being built today, the challenge is to be able to understand their behavior and develop techniques to verify their reliability, security, and safety. This is, indeed, the core motivation of the scientific community around CPS. As a result, CPS has been driven by two mutually reinforcing drivers: applications and theoretical foundations.

Applications

CPS applications have allowed researchers to team up with practitioners to better understand the problems and challenges and provide solutions that can be tested in practical settings. This is the case with medical devices, for example: CPS researchers have teamed up with medical doctors to understand the origin and challenges of medical device-based errors. For instance, some errors associated with infusion pumps were caused by incorrect assumptions about how the human body processes different drugs, how to implement the safeguards to avoid overdose, and how to ensure that a nurse enters the correct information. Furthermore, the current generation of medical devices are certified only as individual devices and are not allowed to be connected to one another. As a result, health practitioners are required to coordinate the use of these devices and ensure the safety of their interactions. For instance, if an X ray of the chest is taken during an operation, it is necessary to ensure that the respirator is disabled (if one is in used). However, once the X rays are taken, the respirator needs to be re-enabled within a safe interval of time to prevent the patient from suffocating. While this invariant could be implemented in software, the current certification techniques and policies prevent this kind of integration from happening. Researchers working in this area are developing techniques to enable the certification of these interactions. This issue is discussed at length in Chapter 1, "Medical Cyber-Physical Systems."

The electric grid is another important application area due to its national importance as a critical infrastructure. A key challenge in this area is the uncoordinated nature of the actions that affect the consumption and generation of electricity by a large number of consumers and providers, respectively. In particular, each individual household is able to change its consumption with the flip of a switch; the consequences of these decisions then have an aggregate effect on the grid that needs to be balanced with the supply. Similarly, renewable power-generation sources, such as wind and solar energy, provide sporadic and unpredictable bursts of energy, making the balance of supply and demand very challenging with such systems. More importantly, the interactions between these elements are both cyber and physical in nature. On the one hand, some level of computer-mediated coordination happens between suppliers. On the other hand, the interactions with the consumer occur mostly through the physical consumption of electricity. Today, a number of techniques are already being used for the development and control of power grids to prevent damage to the infrastructure and improve reliability. However, new challenges require a new combination of cyber and physical elements that can support efficient markets, renewable energy sources, and cheaper energy prices. Chapter 2, "Energy Cyber-Physical Systems," discusses the challenges and advances in the electric grid domain.

Perhaps one of the most interesting application areas that has created its own technical innovations is sensor networks. The development and deployment of sensor networks faces challenges of space, time, energy, and reliability that are very particular to this area. The challenges facing sensor networks as well as the main technical innovations in this area are discussed in Chapter 3, "Cyber-Physical Systems Built on Wireless Sensor Networks."

Other application areas have their own momentum, and yet other emerging areas may soon appear on the horizon. This book discusses some of the most influential areas that are defining the CPS discipline.

Theoretical Foundations

The theoretical advances in CPS have largely focused on the challenges imposed by the interactions between multiple scientific domains. This is the case, for instance, with real-time scheduling. A few trends in this area are worth mentioning. First, new scheduling models to accommodate execution overloads have appeared. These models combine multiple execution budgets with a criticality classification to guarantee that during a normal operation all tasks will meet their deadlines; if an overload occurs, however, the high-criticality tasks meet their deadlines by stealing processor cycles from low-criticality tasks. The second trend comes from variations in periodicity. The so-called *rhythmic task* model allows tasks to continuously vary their periodicity following physical events of variable frequency. This is the case, for instance, with tasks triggered by the angular position of the crankshaft in the engine of a car: New scheduling analysis techniques were developed to verify the timing aspect of these systems. The real-time scheduling foundations and innovations are discussed in Chapter 9, "Real-Time Scheduling for Cyber-Physical Systems."

Cross-cutting innovations between model checking and control theory for control synthesis are among the recent noteworthy developments. In this scheme, hybrid state machine models are used to describe the behavior of the physical plant and the requirements of the computational algorithm. Then, this model is used to automatically synthesize the controller algorithm enforcing the desired specifications. This scheme is discussed at length in Chapter 4, "Symbolic Synthesis for Cyber-Physical Systems." Similarly, a number of new techniques have been developed to analyze the timing effects of a scheduling discipline in control algorithms. These issues are discussed in Chapter 5, "Software and Platform Issues in Feedback Control Systems."

Another area of interaction that has been explored is the relationship between model checking and scheduling. In this case, a new model checker called REK was developed to take advantage of the constraints that the rate-monotonic scheduler and periodic task model imposes on task inter-leavings in an effort to reduce the verification efforts. These new interactions are discussed in Chapter 6, "Logical Correctness for Hybrid Systems."

Security is another big area that is significantly affected by the presence of physical processes. In particular, the interactions between software and physical processes present new opportunities for potential attackers that make CPS security different from software-only security. This difference arises because attacks against, say, sensors to provide false sensor readings can sometimes be very difficult to distinguish from genuine readings from the physical processes. A number of innovations to prevent this kind of man-in-the-middle attack, as well as other significant techniques, are presented in Chapter 7, "Security of Cyber-Physical Systems."

For distributed systems, new techniques to enforce synchronized communication between distributed agents have proved very useful to reduce the effort needed to produce formal proofs of functional correctness in distributed real-time systems. This issue is discussed at length in Chapter 8, "Synchronization in Distributed Cyber-Physical Systems."

CPS analysis techniques rely on models, and the formal semantics of these models is a key challenge that must be addressed. Chapter 10, "Model Integration in Cyber-Physical Systems," presents the latest developments in the definition of formal semantics for models in what are called model-integration languages.

A large number of theoretical advances are discussed in this book, along with the open challenges in each area. While some advances stem from specific challenges in application areas, others expose new opportunities.

Target Audience

This book is aimed at both practitioners and researchers. For practitioners, the book profiles both the current application areas that are benefiting from CPS perspectives and the current techniques that had proved successful for the development of the current generation of CPS. For the researcher, this book provides a survey of application areas and highlights their current achievements and open challenges as well as the current scientific advances and their challenges.

The book is divided into two parts. Part I, "Cyber-Physical System Application Domains," presents the current CPS application areas that are driving the CPS revolution. Part II, "Foundations," presents the current theoretical foundations from the multiple scientific disciplines used in the development of CPS.

This page intentionally left blank

Part I

Cyber-Physical System Application Domains

This page intentionally left blank

Chapter 1

Medical Cyber-Physical Systems[1]

Insup Lee, Anaheed Ayoub, Sanjian Chen, Baekgyu Kim, Andrew King, Alexander Roederer, and Oleg Sokolsky

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, including 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, provides case studies that illustrate these challenges and suggests ways to address them, and highlights several open research and development issues. It concludes with a discussion of the implications of MCPS for stakeholders and practitioners.

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

1.1 Introduction and Motivation

The two most significant transformations in the field of medical devices in recent times are the high degree of reliance on software-defined functionality and the wide availability of network connectivity. The former development means that software plays an 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 patient's physiology. The combination of the embedded software controlling the devices, the new networking capabilities, and the complicated physical dynamics of the human body makes modern medical device systems a distinct class of cyber-physical systems (CPS).

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, embedded technologies and CPS. One of the primary concerns in MCPS development is the assurance of patient safety. The new capabilities of future medical devices and the new techniques for developing MCPS with these devices will, in turn, 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 lengthy and prohibitively expensive owing to the increased MCPS complexity, and there is an urgent need to ease this often onerous process without compromising the level of safety it delivers.

In this chapter, we advocate a systematic approach to analysis and design of MCPS for coping with their inherent complexity. Consequently, we suggest that 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, well 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 during the implementation stage. 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.

This chapter describes some of the research directions being taken to address the various challenges involved in building MCPS:

- *Stand-alone device:* A model-based high-assurance software development scheme is described for stand-alone medical devices such as patient-controlled analgesia (PCA) pumps and pacemakers.
- *Device interconnection:* A medical device interoperability framework is presented for describing, instantiating, and validating clinical interaction scenarios.
- *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.
- *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.
- *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.

MCPS are viewed in a bottom-up manner in this chapter. That is, we first describe issues associated with individual devices, and then progressively increase their complexity by adding communication, intelligence, and feedback control. Preliminary discussion of some of these challenges has appeared in [Lee12].

1.2 System Description and Operational Scenarios

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 which treatment options can be chosen and which adjustments of treatment settings need to be made in response to changes in the patient's condition.

Traditionally, decisions about the treatment options and settings have been made by the attending caregiver, who makes them by monitoring patient state using individual devices and performing manual adjustments. Thus, clinical scenarios can be viewed as closed-loop systems in which 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 1.1 presents a 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
- *Delivery devices*, such as infusion pumps and ventilators, which actuate therapy that is capable of changing the patient's physiological state

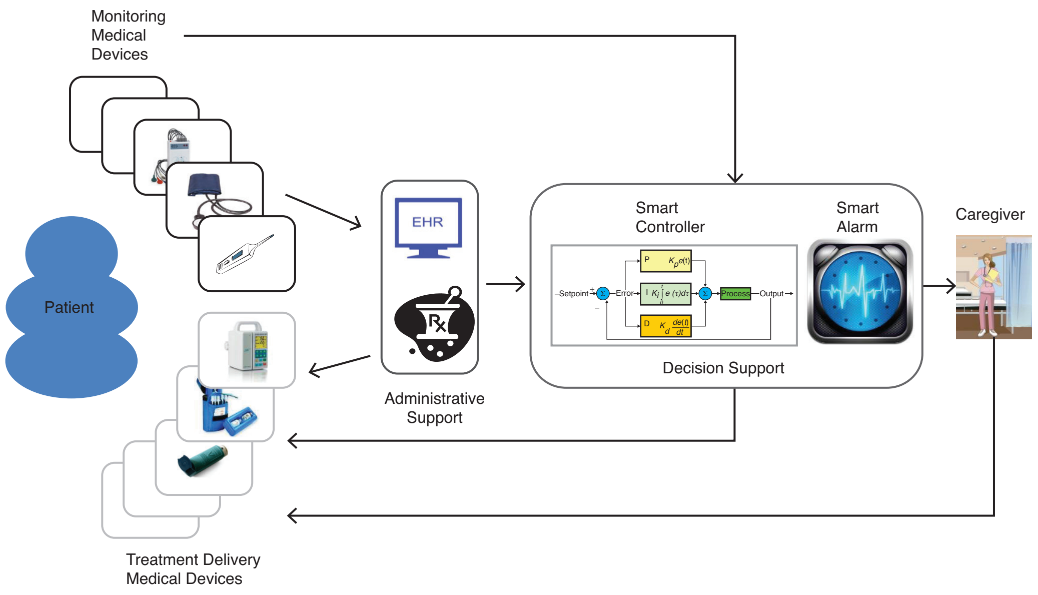


Figure 1.1: *A conceptual overview of medical cyber-physical systems*

© 2012 IEEE. Reprinted, with permission, from *Proceedings of the IEEE* (vol. 100, no. 1, January 2012).

In MCPS, interconnected monitoring devices can feed collected data to decision support or administrative support entities, each of which serves a different, albeit complementary, purpose. For example, caregivers can analyze the information provided by these devices and then use delivery devices to initiate treatment, thereby bringing the caregiver into the control loop around the patient. Alternatively, the decision support entities might 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 the safety of these devices and their interoperation is crucial. One of the more effective strategies to do so is to use model-based development methods, which can ensure device safety by enabling medical device verification. This strategy also opens the door for eventually certifying the devices to meet certain safety standards.

1.2.1 Virtual Medical Devices

Given the high complexity of MCPS, any such system has to be user-centric; that is, it must be easy to set up and use, in a largely automated manner. One way to accomplish this is to develop a description of the MCPS workflow and then enforce it on physical devices. MCPS workflow can be described in terms of the number and types of devices involved, their mutual interconnections, and the clinical supervisory algorithm needed for coordination and analysis of data collected by the system. Such a description defines *virtual medical device* (VMD). VMDs are used by a *VMD app* and instantiated during the setup of actual medical devices—that is, as part of a *virtual medical device instance*.

The devices in a 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, therefore, is to find the medical devices in a 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 the 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 according to the specification. Once the workflow has run its course, the VMD app can perform the necessary cleanup to allow another workflow to be specified using a different combination of medical devices in the VMD instance.

1.2.2 Clinical Scenarios

Each VMD supports 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 and ventilator coordination and another for a patient-controlled analgesia (PCA) safety interlock system.

One example that illustrates how patient safety can be improved by MCPS is the development of a VMD that coordinates the interaction between an X-ray machine and a ventilator. Consider the scenario described by [Lofsky04]. X-ray images are often taken during surgical procedures. If the surgery is being performed under general anesthesia, the patient breathes with the help of a ventilator during the procedure. Because the patient on ventilator cannot hold his or her 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. In some unfortunate cases, 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, albeit one 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, between the time when the patient has finished exhaling and the time he or she starts 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 decide 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 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 distinct dosages and delivery schedules. PCA pumps allow patients to press a button to request a dose when they decide they want it, rather than using a dosing schedule fixed by a caregiver. Some patients may decide they prefer a higher level of pain to the nausea that the drugs may cause and, therefore, press the button less often; others, who need a higher dose, can press the button 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 prevent an overdose by limiting how many doses it will deliver, regardless of how often the patient pushes the button. However, this safety mechanism is not sufficient to protect all patients. Some patients may still receive overdoses if the pump is misprogrammed, if the pump programmer overestimates the maximum dose that 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 associated with 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].

1.3 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 are raising new challenges in the development of high-confidence medical devices, yet are simultaneously opening up new opportunities for the research community [Lee06]. This section begins by reviewing the main trends that have emerged recently, then identifies quality attributes and challenges, and finally provides a detailed discussion of several MCPS-specific topics.

1.3.1 Trends

Four trends in MCPS are critical in the evolution of the field: software as the main driver of new features, device interconnection, closed loops that automatically adjust to physiological response, and a new focus on continuous monitoring and care. The following subsections discuss each of these trends.

1.3.1.1 New Software-Enabled Functionality

Following the general trend in the field of embedded systems, and more broadly in cyber-physical systems, introduction of 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. One of the most technology-intensive medical procedures, it requires one of the largest-scale medical device systems. To deliver its precise doses of radiation to patients with cancer, the treatment requires precise guiding of a proton beam from a cyclotron to patients, but must be able to adapt to even minor shifts in the patient's 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 locations in the patient's body and needs to be switched from location to location, opening up the possibility of interference between beam scheduling and application. In addition to controlling the proton beam, a highly critical function of software in a proton treatment system is real-time image processing to determine the precise position of the patient and detect any patient movement. In [Rae03], the authors analyzed the safety of proton therapy machines, but their analysis concentrated on a single system, the emergency shutdown. In general, proper analysis and validation of such large and complex systems remains one of the biggest challenges facing the medical device industry.

As further evidence of the software-enabled functionality trend, even in simpler devices, such as pacemakers and infusion pumps, more and more software-based features are being added, making their device software more complex and error prone [Jeroeno4]. Rigorous approaches are required to make sure that the 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].

1.3.1.2 Increased Connectivity of Medical Devices

In addition to relying on software to a greater extent, medical devices are increasingly being equipped with network interfaces. In essence, interconnected medical devices form a distributed medical device system of a larger scale and complexity that must be properly designed and validated to ensure effectiveness and patient safety. Today, the networking capabilities of medical devices are primarily exploited for patient monitoring purposes (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. The 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, with the ultimate goal of improving patient safety and health care efficiency. In addition to developing interoperability standards, the MD PnP initiative collects and demonstrates clinical scenarios in which interoperability leads to improvement over the existing practice.

1.3.1.3 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 a surgical procedure 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 humans being in the loop may compromise patient safety. Caregivers, who are often overworked and operate under severe time pressures, may miss a critical warning sign. Nurses, for example, typically care for multiple patients at a time and can become distracted. Using an automatic controller to provide continuous monitoring of the patient state and handling of routine situations would relieve some of the pressure on the caregiver and might potentially 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—for example, 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 into mainstream practice as yet.

1.3.1.4 Continuous Monitoring and Care

Due to the high costs 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, 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 are becoming more popular. 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.

1.3.2 Quality Attributes and Challenges of the MCPS Domain

Building MCPS applications requires ensuring the following quality attributes, which in turn pose significant challenges:

- *Safety:* Software is playing 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 ensure the safety and effectiveness of MCPS. We advocate the use of model-based development and analysis as a means of ensuring the 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 such systems is the concept of interoperability, wherein 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, and 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, with the combined data then being used to enable early detection of ailments and generate effective alarms in the event of emergency. However, given the complexity of human physiology and the many variations of physiological parameters over patient populations, developing such computational intelligence is a nontrivial task.
- *Autonomy:* The computational intelligence that MCPS possess can be applied to increase 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 major challenge, primarily due to the complexity and variability of human physiology.
- *Security and privacy:* Medical data collected and managed by MCPS are 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 by exchanging patient data from multiple sources; however, it also increases the vulnerability of the system to security and privacy violations.
- *Certification:* A report by the U.S. National Academy of Science, titled "Software for Dependable Systems: Sufficient Evidence?," recommends an 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, therefore, is both 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 that provides a convincing and consistent argument that a system is adequately safe (or secure) [Menon09]. The notion of assurance cases holds the promise of providing an objective, evidence-based approach to software certification. Assurance cases are increasingly being used as a means

of 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.

1.3.3 High-Confidence Development of MCPS

The extreme market pressures faced by the medical devices industry has forced many companies to reduce their 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. Model-based development can be a significant part of such a 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. Nevertheless, the process itself is general enough to accommodate a wide variety of rigorous development technologies.

1.3.3.1 Mitigation of Hazards

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

Figure 1.2 depicts a relatively conventional approach to high-assurance development of safety-critical systems based on the mitigation of hazards. The process starts with the identification of the desired functionality and the hazards associated with the system's operation. The chosen functionality yields the system functional requirements, while hazard mitigation strategies yield the system safety requirements.

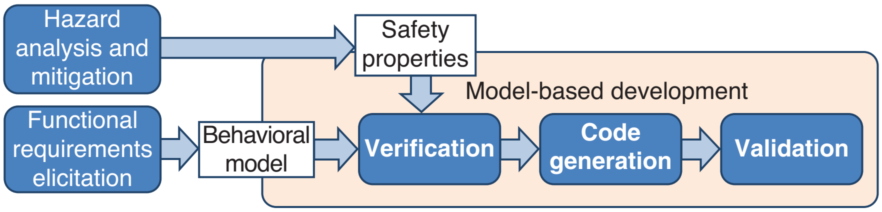


Figure 1.2: *High-assurance development process for embedded software*

The functional requirements are used to build detailed behavioral models of the software modules, while the 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 consists of verification, code generation, and validation phases.

Model-based development has emerged as a means of raising the level of assurance in software systems. In this approach, developers start with declarative models of the system and perform a rigorous model verification with respect to safety and functional requirements; they then use systematic code generation techniques to derive code that preserves the verified properties of the model. Such a development process allows the developers to detect problems with the design and fix them at the model level, early in the design cycle, when 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], so they do not allow developers to fully utilize the benefits of model-based design. The use of formal modeling facilitates making mathematically sound conclusions about the models and generating code from them.

1.3.3.2 Challenges of Model-Driven Development of MCPS

Several challenges arise when developing 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, but 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. Conversely, a very detailed model makes code generation relatively straightforward, but pushes the limits of the currently available verification tools.

Many modeling approaches rely on the separation of the platform-independent and platform-dependent aspects of development. 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 another level of complexity to the model, which may increase verification time to an unacceptable duration.

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, such as that implemented on many infusion pumps. Some pumps may not have a physical sensor for that purpose and simply 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. At the same time, such separation leads to integration challenges at the implementation level. The code generated by the platform-independent model needs to be integrated with the code from the various target platforms in such a way that the verified properties of the platform-independent model are preserved.

Second, there is often a semantic gap between the model and the 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 the 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 by using instantaneous channel synchronization or event broadcasting that has a zero time delay. Such semantics simplifies modeling input and output of the system so that the modeling/verification complexity is reduced. Unfortunately, the correct implementation of such semantics is hardly realizable at the implementation level, because execution of those actions requires interactions among components that have a non-zero time delay.

The following case study concentrates on the development of a PCA infusion pump system and considers several approaches to address these challenges.

1.3.3.3 Case Study: PCA Infusion Pumps

A PCA infusion pump primarily delivers pain relievers, and is equipped with a feature that allows for additional limited delivery of medication, called a bolus, upon patient demand. This type of infusion pump is widely used for pain control of postoperative patients. If the pump overdoses opioid drugs, however, the patient can be at risk of respiratory depression and death. Therefore, these medical devices are subject to stringent safety requirements that aim to prevent overdose.

According to the FDA's Infusion Pump Improvement Initiative [FDA10a], the FDA received more than 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 Generic PCA Project

The Generic PCA (GPCA) project, a joint effort between the PRECISE Center at the University of Pennsylvania and researchers at the FDA, aims to develop a series of publicly available artifacts that can be used as guidance for manufacturers of PCA infusion pumps. 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, companies 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 this effort is presented in [Kim11].

The development approach follows the process outlined in Figure 1.2. The detailed steps are shown in Figure 1.3. 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 with its safety requirements. The assurance case development is discussed in more detail in Section 1.3.7.

Modeling

The reference model of the GPCA pump implemented in Simulink/Stateflow is used as the source of functional requirements and converted to UPPAAL [Behrmann04] via a manual but systematic translation process. The model structure follows the overall architecture of the reference model, which is shown in Figure 1.4. 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.

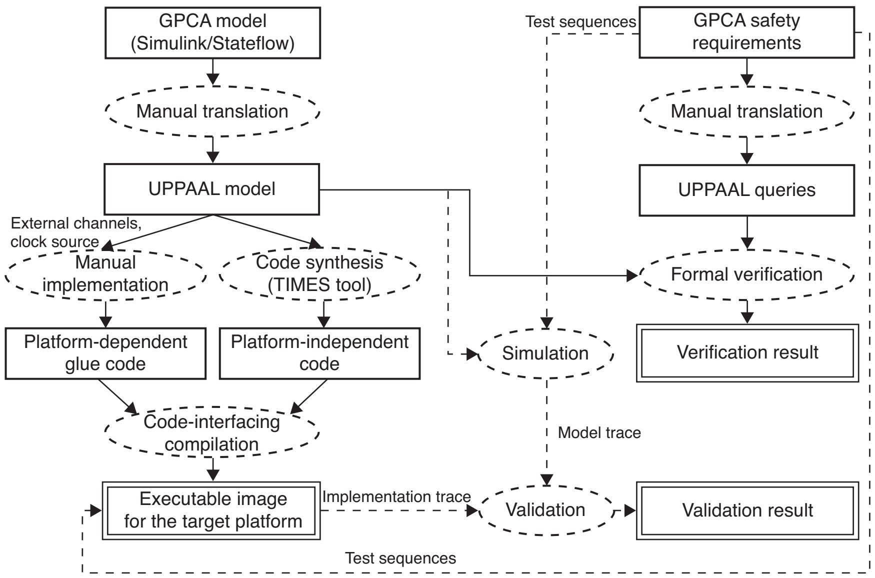


Figure 1.3: *The model-driven development for the GPCA prototype*

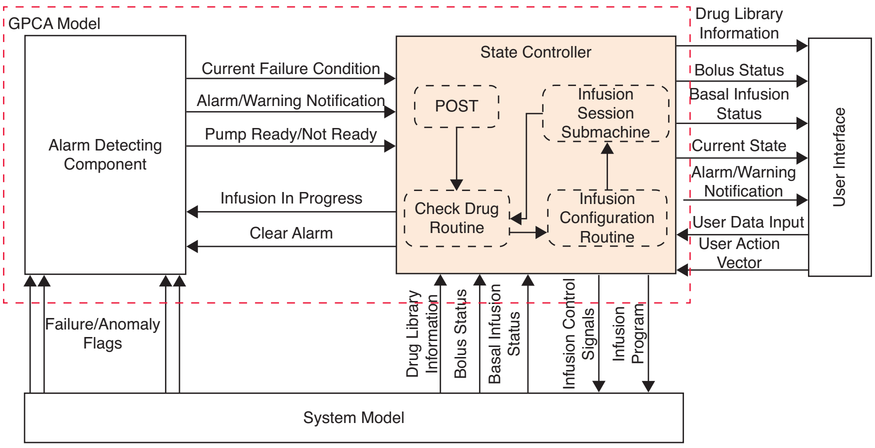


Figure 1.4: *The system architecture of the GPCA model*

The state machines are organized as a set of modes, with each mode captured as a separate submachine. In particular, the state controller contains four modes:

- Power-on self-test (POST) mode is the initial mode that checks system components on start-up.

- The check-drug mode represents a series of checks that the caregiver performs to validate the drug loaded into the pump.
- The infusion configuration mode represents interactions with the caregiver to configure infusion parameters such as infusion rate and volume to be infused (VTBI) and validate them against the limits encoded in the drug library.
- The infusion session is where the pump controls delivery of the drug according to the configuration and the 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 can be performed, requirements need to be formalized as properties to be checked. We can categorize the requirements according to their precision and level of abstraction:

- *Category A:* Requirements that are detailed enough to be formalized and verified against the model
- *Category B:* Requirements that are beyond the scope of the model
- *Category C:* Requirements that are too imprecise to be formalized

Only requirements in Category A can be readily used in verification. Just 20 out of the 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. Thus, 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 approach 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—a distinction that is useful in the model-based development. An alternative approach is to handle these requirements outside of the model-based process—for example, validating by testing. In this case, however, the benefits of formal modeling are lost.

A better approach is to match the level of detail by further decomposing the requirements. At the platform-independent level, we might check 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, we might 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 requirement from Category C is "Flow discontinuity at low flows should be minimal," which does not specify what is a low flow or which discontinuity can be accepted as minimal. This case 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 computation tree logic (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 a property-reserving manner. An example of such a 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 triggering of the occlusion alarm from the pump hardware) are abstracted as instantaneous transitions subject to input/output synchronization with their environment. On a particular platform, the underlying operating system schedules the interactions, thereby affecting the timing of their execution.

Several approaches may be used to address this issue at the integration stage. In [Henzinger07], higher-level programming abstraction is proposed as a means to model the timing aspects and generate code that is independent from the scheduling algorithms of a particular platform. The platform integration is then performed by verifying time-safety—that is, checking whether 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 a 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, inevitably some assumptions will be 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 systematically exercises the code using test cases derived from the model. A rich literature on model-based test generation exists; 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.

1.3.4 On-Demand Medical Devices and Assured Safety

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 actively under way. The projects described in this section represent a first step toward understanding the engineering and regulatory challenges associated with such systems. The success and safety of these systems will depend not only on new engineering techniques, but also on new approaches to regulation and the willingness of industry members to adopt appropriate interoperability standards.

1.3.4.1 Device Coordination

Historically, medical devices have been used as individual tools for patient therapy. 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, and prone to errors and accidents.

One example of manual device coordination in current practice is the X ray and ventilator coordination mentioned in Section 1.2; another example is trachea or larynx surgery performed with a laser scalpel. In this type of surgery, the patient is placed under general anesthesia while the surgeon makes cuts on the throat using a high-intensity laser. Because the patient is under anesthesia, his or her breathing is supported by an anesthesia ventilator that 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 rapid combustion, burning the patient from the inside out. To mitigate this hazard, the surgeon and the anesthesiologist must be in constant communication: 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 level drops too low, the anesthesiologist signals the surgeon to stop cutting so oxygen can be supplied again.

If medical devices could coordinate their actions, then the surgeon and the anesthesiologist would not have to expend their concentration and effort to ensure that the activities of the medical devices are safely synchronized. Furthermore, the patient would not be exposed to the potential for human error.

Many other clinical scenarios might benefit from this kind of automated medical device coordination. These scenarios involve either *device synchronization*, *data fusion*, or *closed-loop control*. The laser scalpel ventilator safety interlock epitomizes device synchronization: Each device must always be in a correct state relative to the other devices. In data fusion, physiologic readings from multiple separate devices are considered as a collective. Examples of such applications include smart alarms and clinical decision support systems (see Section 1.3.5). Finally, closed-loop control of therapy can be achieved by collecting data from devices that sense the patient's physiological state and then using those data to control actuators such as infusion pumps (see Section 1.3.6).

1.3.4.2 Definition: Virtual Medical Devices

Let us now clarify the concept of virtual medical devices, including why they are considered a different entity. 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 particular set of devices for the VMD and connects them together.

1.3.4.3 Standards and Regulations

Several existing standards are designed to enable medical device interconnectivity and interoperability. These standards include the Health Level 7 standards [Dolin06], IEEE-11073 [Clarke07, ISO/IEEE11073], 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 the inter-device coordination and control needed with the laser scalpel and ventilator combination. 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. Virtual medical devices, in contrast, 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, or 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 VMDs. Should regulators mandate specific standards? Do regulators need to adopt component-wise certification regimes? What is the role, if any, of third-party certifiers?

1.3.4.4 Case Studies

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-and-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, 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 healthcare organization separately [Hatcliff12].

Integrated Clinical Environment
Figure 1.5 shows the primary components of the integrated clinical environment (ICE) architecture. This case study summarizes the intended functionality and goals for each of these components. Note

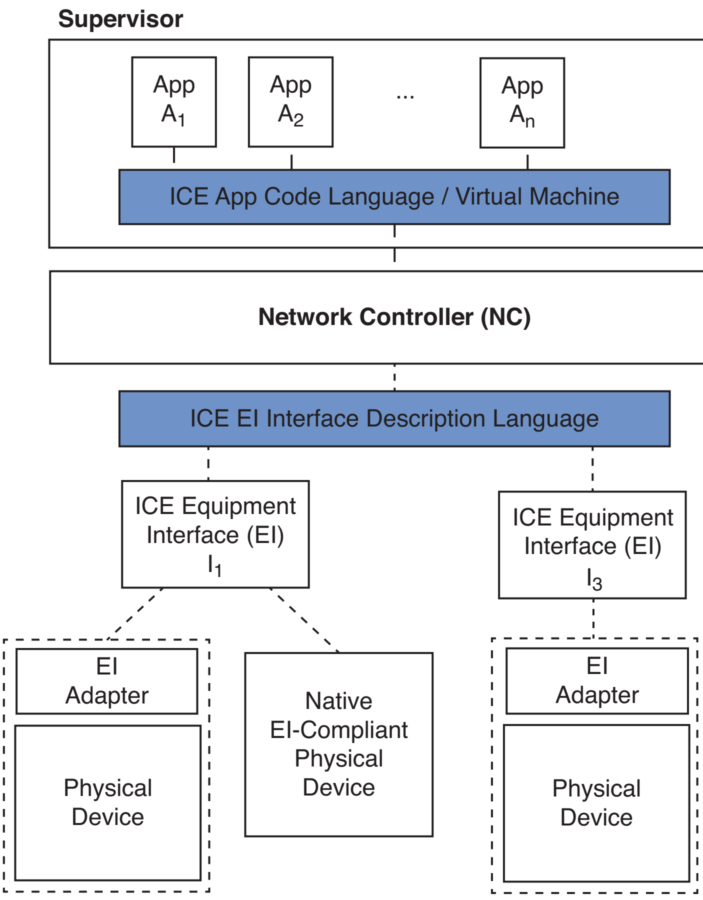

Supervisor
App A_1
App A_2
...
App A_n
ICE App Code Language / Virtual Machine
Network Controller (NC)
ICE EI Interface Description Language
ICE Equipment Interface (EI) I_1
ICE Equipment Interface (EI) I_3
EI Adapter
Physical Device
Native EI-Compliant Physical Device
EI Adapter
Physical Device

Figure 1.5: *ICE architecture*

that ASTM F2761-2009 does not provide detailed requirements for these components, as it is purely an architectural standard. Nevertheless, 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). In addition to executable code, these applications contain device requirements declarations—that is, 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 used in the ICE architecture 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 which sensors and actuators are present on the device, and which command set it supports.

Medical Device Coordination Framework

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 modular framework is envisioned as enabling researchers to rapidly prototype systems and explore implementation and engineering issues associated with on-demand medical systems.

The MDCF is implemented as a collection of services that work together to provide some of the capabilities required by ICE as essential for a medical application platform. The functionality of these services also may be decomposed along the architectural boundaries defined in the ICE architecture (see Figure 1.6); that is, 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, exchanges of 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

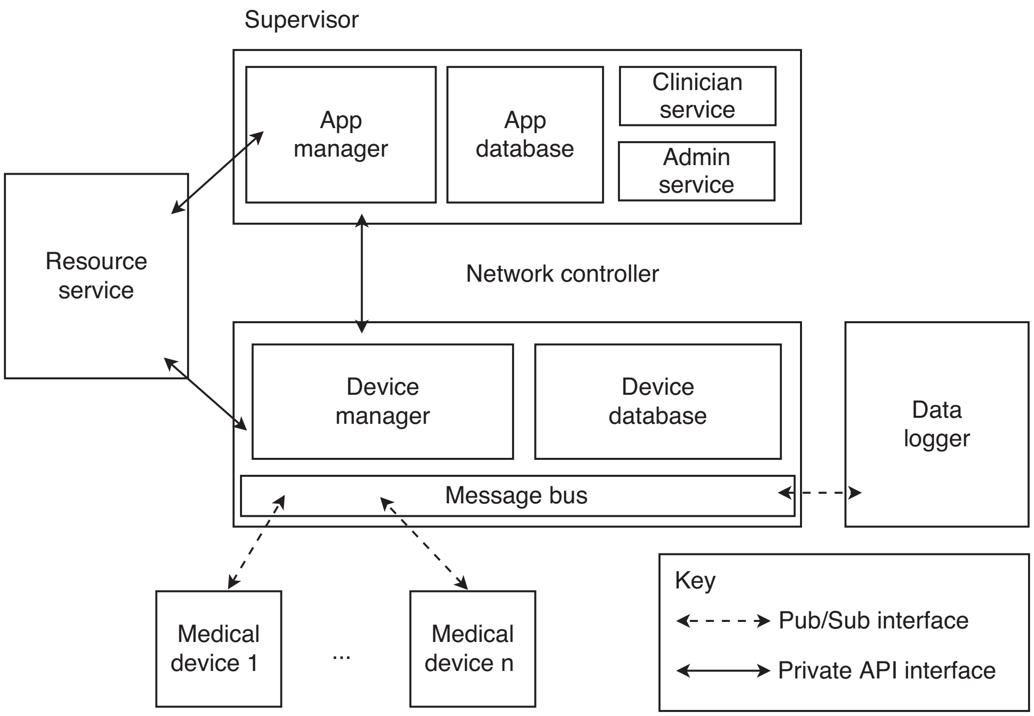


Figure 1.6: *MDCF services decomposed along ICE architectural boundaries*

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 also serves another important role: It validates the trustworthiness of any connecting device by determining whether 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 (e.g., 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 and tamper evident; 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 because 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 so as 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 which 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 GUI 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 it can be run 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.

1.3.5 Smart Alarms and Clinical Decision Support Systems

Fundamentally, clinical decision support (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 test values, and patient history; they then subject those inputs 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.

As more medical devices become capable of recording continuous vital signs, and as medical systems become increasingly interoperable, CDS systems will evolve into essential tools that allow clinicians to process, interpret, and analyze patient data. While widespread adoption of CDS systems in clinical environments faces some challenges, the current efforts to build these systems promise to expose their clinical utility and provide impetus for overcoming those challenges.

1.3.5.1 The Noisy Intensive Care Environment

Hospital intensive care units (ICUs) utilize a wide array of medical devices in patient care. A subset of these medical devices comprises sensors that 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, and so on. These sensors range from very simple to very complex in terms of their technology. Additionally, along with the traditional techniques, digital technologies have enabled new sensors to be developed and evaluated for clinical use.

The vast majority of these medical devices act 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. Each of the signals is still displayed independently, however, so it is up to the clinician to synthesize the presented information to determine the patient's actual condition.

Many of these devices can be configured to alert clinicians to a deterioration in the patient's condition. Most sensors currently in use can be configured with only threshold alarms, which activate when the particular vital sign being measured crosses a predefined threshold. While threshold alarms can certainly be critical 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], often attributable to insignificant random fluctuations in the patient's vital signs or noise caused by external stimuli. For example, patient movement can cause sensors to move, be compressed, or fall off. The large number of erroneous alarms generated by such devices causes alarm fatigue—a desensitization to the presence of these alarms that causes clinicians to ignore them [Commission13]. In an effort to reduce the number of false alarms, clinicians may sometimes improperly readjust settings on the monitor or turn off alarms entirely [Edworthy06]. Both of these actions can lead to missed true alarms and a decrease in quality of care [Clinical07, Donchin02, Imhoff06].

Various efforts have been made to reduce alarm fatigue. These strategies 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 simply 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 collection of findings can be cause for concern. Thus, it seems pertinent to develop smart alarm systems that would consider multiple vital signs in concert before raising an alarm. This would reduce false alarms, improving the alarm precision and reducing alarm fatigue, thereby leading to improved care.

Such a smart alarm system would be a simple version of a 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 CDS 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.

1.3.5.2 Core Feature Difficulties

As CDS systems are a specialized form of MCPS, the development of CDS systems 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 in part reflects 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 the real-time vital signs data being collected by a number of different medical devices attached to the patient. To obtain these data, the devices collecting the required vital signs must be able to interoperate—if not with each other, then with a central data repository. In this repository, data could be collected, time synchronized, analyzed, and visualized.

In the past, achieving interoperability of medical devices has been a major hurdle. Due to increased costs, the exponential increase in regulatory difficulty, and the lucrative potential from selling a suite of devices with limited interoperability, individual device manufacturers currently have few incentives to make their devices interoperate. Development of an interoperable 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 depend on other factors, such as network reliability and real-time guarantees for 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 related to 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, therefore, is a vital component of the development of CDS systems. Addressing this problem will require hospitals to recognize the value of capturing and storing patients' data and to develop a 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 clinicians' mental models, and by creating models based on machine learning of medical data.

While the majority of hospital guidelines can usually be encoded as a series of simple rules, they are often vague or incomplete. Thus, while they may serve as a useful baseline, such guidelines are often insufficient on their own to realize context-aware computational intelligence. Capturing clinicians' mental models involves interviewing a large number of clinicians about their decision-making processes and then hand-building an algorithm based on the knowledge gleaned from the interviews. This process can be laborious, it can be difficult to quantify in software how a clinician thinks, 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 sets are available, they often prove to be noisy, with many missing values. The choice of learning technique can be a difficult one, too. 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.

1.3.5.3 Case Study: A Smart Alarm System for CABG Patients

Patients who have undergone coronary artery bypass graft (CABG) surgery are at particular risk of physiologic instability, so continuous monitoring of their vital signs is routine practice. The hope is that detection of physiologic changes will allow practitioners to intervene in a timely manner and prevent postsurgery complications. As previously discussed, the continuous vital signs monitors are usually equipped only with simple threshold-based alarms, which, in combination with the rapidly evolving post-surgical state of such patients, can lead to a large number of false alarms. For example, it is common for the finger-clip sensors attached to pulse oximeters to fall off 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 that combines four main vital signs routinely collected in the surgical ICU (SICU): blood pressure (BP), heart rate (HR), respiratory rate (RR), and blood oxygen saturation (SpO_2). ICU nurses were interviewed to determine appropriate ranges for binning each vital sign into a number of ordinal sets (e.g., "low," "normal," "high," and "very high," leading to classifying, for example, a blood pressure greater than 107 mm Hg as "high"). Binning vital signs in this way helped overcome the difficulty of establishing a rule set 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 was developed in conjunction with nurses to identify combinations of these vital signs statuses that would be cause for concern. The smart alarm monitors a patient's four vital signs, categorizes them according to which ordinal set they belong in, 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 for a vital sign are conservatively classified as "low" for the duration of the signal drop.

This smart alarm avoided many of the challenges that CDS systems normally face in the clinical environment. The set of vital signs employed was very limited and included only those 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 it 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" provided a conservative fallback in case of a brief network failure. Additionally, running the system on a real-time middleware product would provide the necessary data delivery guarantees to ensure system safety.

To evaluate the performance of this system, 27 patients were observed while they convalesced in the ICU immediately after their CABG procedure. Of these 27 patients, 9 had the requisite vital signs 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. 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 data store. 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 bedside. 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 10 interventions during the observation time period, 9 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 rather by their trend. An improved version of this smart alarm system would include rules concerning the trend of each of the vital signs.

1.3.6 Closed-Loop System

Given that medical devices are aimed at controlling a specific physiological process in a human, they can be viewed as a closed loop between the device and the patient. In this section, we discuss clinical scenarios from this point of view.

1.3.6.1 A Higher Level of Intelligence

A clinical scenario can be viewed as a control loop: The patient is the plant, and 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. This role imposes a 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, whereby the patient's condition is under constant surveillance, is an active area of research [Maddox08]. However, to improve patient safety further, the system should be able to continuously react to changes in patient condition as well.

The smart alarm systems and decision support systems, discussed in the previous section, facilitate the integration and interpretation of clinical information, helping caregivers make decisions 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—for example, autopilots in airplanes and adaptive cruise control in vehicles. In patient care, the controller can continuously monitor the patient's state and automatically reconfigure the actuators when the patient's condition stays within a predefined operation region. It will alert and hand control back to caregivers if the patient's state starts veering out of the safe range. Such physiological closed-loop systems can assume part of the caregivers' workload, enabling them to 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 to apply, which may improve both the safety and the effectiveness of patient care.

The concept of closed-loop control has already been introduced in medical applications—for example, in implantable devices such as cardioverter defibrillators and other special-purpose stand-alone 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.

1.3.6.2 Hazards of Closed-Loop Systems

The networked closed-loop setting introduces new hazards that could compromise patient safety. These hazards need to be identified and mitigated in a systematic way. Closed-loop MCPS, in particular, 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 biomedical engineers and medical experts, and the area remains 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 nonlinear and contain parameters that are highly individual dependent, time varying, and not easily identifiable given the technologies available. This imposes a major burden on control design as well as system-level safety reasoning.

Second, in the closed-loop medical device system, complex interactions occur 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 must be considered in the safety arguments.

Third, the control loop is subject to uncertainties caused by sensors, actuators, and communication networks. For example, some body sensors are very sensitive to patient movements—vital signs monitors may alert faulty readings due to a dropped finger-clip—and due to technological constraints, some biosensors 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 as they travel across the network.

1.3.6.3 Case Study: Closed-Loop PCA Infusion Pump

One way to systematically address the challenges faced by closed-loop systems is to employ a model-based approach similar to the one outlined in Section 1.3.3. This effort involves extending the high-confidence approach based on hazard identification and mitigation from individual devices to a system composed of a collection of devices and a patient.

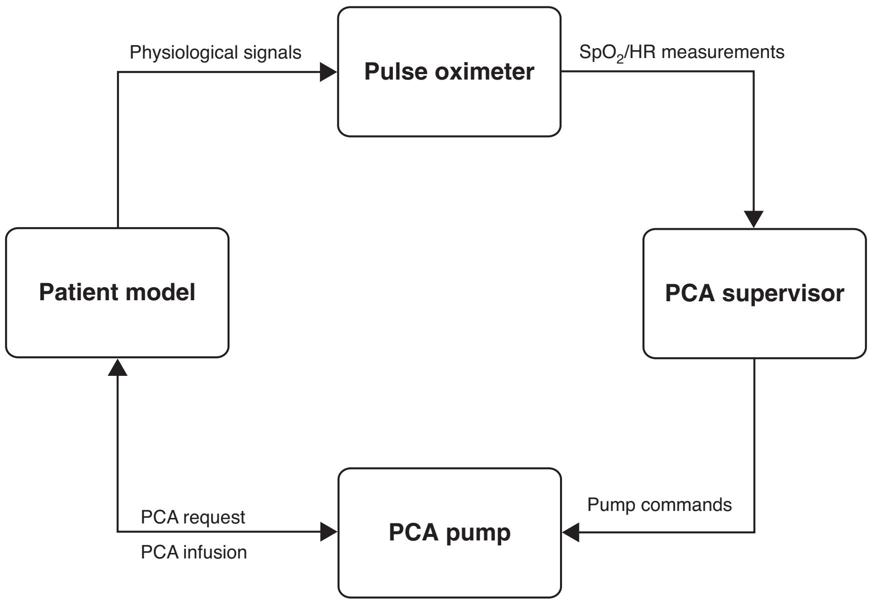
This section briefly describes a case study of the use of physiological closed loop in pain control using a PCA infusion pump, introduced in Section 1.3.3.3. The biggest safety concern that arises with the use of PCA pumps for pain control is the risk of overdose of an opioid analgesic, which can cause respiratory failure. Existing safety mechanisms built into PCA pumps include limits on bolus amounts, which are programmed by a caregiver before the infusion starts, and minimum time intervals between consecutive bolus doses. In addition, nursing manuals prescribe periodic checks of the patient condition by a nurse, although 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 1.3.4. The pulse oximeter continuously monitors heart rate and blood oxygen saturation. The controller receives measurements from the pulse oximeter, and it may stop the PCA infusion if the HR/SpO_2 readings indicate a dangerous decrease in respiratory activity, thereby preventing overdosing.

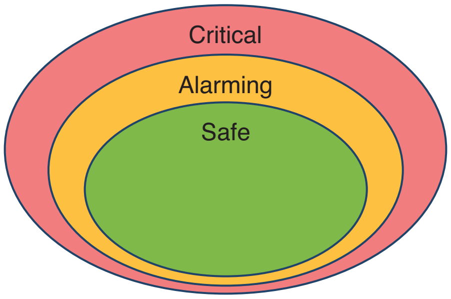
Safety requirements for this system are based on two regions in the space of possible patient states as reported by the two sensors, as illustrated in Figure 1.7. The critical region represents imminent danger to the patient and must be avoided at all times; the alarming region is not immediately dangerous but raises clinical concerns.

The control policy for the safety interlock may be to stop the 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, so as to avoid false alarms that would 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.

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, the pulse oximeter, the control algorithm, and the physiology of the patient.



(a) Closed-loop PCA system



(b) Regions of patient's possible conditions

Figure 1.7: *PCA safety interlock design*

Patient modeling is the critical aspect in this case. Both pharmacokinetic and pharmacodynamics aspects of physiology should be considered [Mazoit07]. Pharmacokinetics specifies how the internal state of the patient, represented by the drug concentration in the blood, is affected by the rate of infusion. Pharmacodynamics specifies how the patient's 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 the patient's vital signs was assumed.

Verification efforts concentrated on the timing of the control loop. After a 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 the signal to the pump, and finally stopping the pump motor. To strengthen confidence in the verification results, the continuous dynamics of the patient model were 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. With this approach, the verification can 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}.

1.3.6.4 Additional Challenging Factors

The PCA system is a relatively simple but useful use case of closed-loop medical devices. Other types of closed-loop systems, by comparison, may introduce new engineering challenges due to their functionalities and requirements. For example, blood glucose control for patients with diabetes has garnered a lot of attention from both the engineering and clinical communities, and various concepts of closed-loop or semi-closed-loop systems have been proposed [Cobelli09, Hovorka04, Kovatchev09]. Compared to the PCA system, the closed-loop glucose control system is substantially more complex and opens up 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, stopping the infusion is considered a safe action, at least for a reasonable time duration. This 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. In this case, stopping the insulin pump is not a default safe action, because high glucose level is also harmful.

The safety criteria in the PCA system are defined by delineating a region in the state space of the patient model (such as the critical region in the previous case study). 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 a certain level of resilience, and the true relationship 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 lower-level concentration that persists over a longer interval.

The pulse oximeter—the sensor used in the PCA system—is relatively accurate with respect to the ranges that clinicians would consider in their decision making. In some other scenarios, however, sensor accuracy is a non-negligible factor. For example, a glucose sensor can have a relative error of as much as 15% [Ginsberg09]; given that the target range is relatively narrow, such an 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, for example, this value may not decline until a relatively late point, after harm to the patient is already done. Capnography data, which measure levels of carbon dioxide exhaled by the patient, can be used to detect the problem much sooner, but this technique is more expensive and involves invasive technology compared to pulse oximetry. This example highlights 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 are alerted when certain conditions arise, 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, however, the user may demand a more hands-on 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; even when the automatic controller is running, the user may choose to reject certain control actions for various reasons (e.g., the patient is not comfortable with a large insulin dose). This kind of more complicated user interaction pattern introduces new challenges to the model-based validation and verification efforts.

1.3.7 Assurance Cases

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 devices domain, the FDA issued draft guidance for medical infusion pump manufacturers indicating that they should provide a safety case with their premarket submissions [FDA10]. In this section, we briefly introduce the concept of safety cases and the notations used to describe them. Three aspects of safety cases that can be manipulated to make them practically useful are discussed—namely, facilitating safety case 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 the period in which a device's application is in FDA-approval limbo. Qualitative reasoning for having confidence in a device is believed to be more consistent with the inherited subjectivity in safety cases than the quantitative 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 [Ayoub13, Cyra08, Kelly07].

Given the subjective nature of safety cases, the review methods cannot hope to replace the human reviewer. Instead, they form frameworks that lead safety case reviewers through the evaluation process. Consequently, the result of the safety case review process is always subjective.

1.3.7.1 Safety Assurance Cases

The safety of medical systems is of great public concern—a concern that is reflected in the fact that many such systems must adhere to government regulations or be certified by licensing bodies [Isaksen97]. For example, medical devices sold in the United States are regulated by the 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, 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 when used in a given context [Menon09]. The notion of safety cases is currently embraced by several European industry sectors (e.g., aircraft, trains, nuclear power). More recently in the United States, the FDA issued draft guidance indicating that medical infusion pump manufacturers should provide a safety case with their premarket submissions [FDA10]. Thus, an infusion pump manufacturer is 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 showing that its product is acceptably safe to operate in the intended context [Kelly98]. The regulator's role, in turn, is to assess the submitted safety case and make sure that the system is really safe.

Many different approaches are possible for the organization and presentation of safety cases. Goal Structuring Notation (GSN) is one description technique that has proved 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—that is, claims about the system specified with text within rectangular elements—are supported by valid and convincing arguments. To this end, goals are successively decomposed into subgoals through implicit or explicit strategies. Strategies, specified with text within parallelograms, explicitly define how goals are decomposed into subgoals. 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 rectangles with rounded sides.

Another popular description technique is called Claims–Arguments–Evidence (CAE) notation [Adelard13]. While this 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.

1.3.7.2 Justification and Confidence

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

All three aspects of assurance cases bring own challenges. These challenges need to be addressed to make safety cases practically useful:

- *Building assurance cases:* The *Six-Step method* [Kelly98a] is a widely used method for systematically constructing safety cases. Following the Six-Step method or any other method does not prevent safety case developers from making some common mistakes, such as leaping from claims to evidence. Even so, capturing successful (i.e., convincing, sound) arguments used in safety cases and reusing them in constructing new safety cases can minimize the mistakes that may be made during the safety case development. The need for argument reusability motivates the use of the pattern concept (where *pattern* means a model or original used as an archetype) in the safety case constructions. Predefined patterns can often provide an inspiration or a starting point for new safety case developments. Using patterns may also help improve the maturity and completeness of safety cases. Consequently, patterns can help medical device manufacturers to construct safety cases in a more efficient way in terms of completeness, thereby shortening the development period. The concept of safety case patterns is defined in [Kelly97] as a way to capture and reuse "best practices" in safety cases. Best practices incorporate company expertise, successfully certified approaches, and other recognized means of assuring quality. 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, Ayoub12, Hawkins09, Kelly98, Wagner10, Weaver03] to capture best practices.

- *Trusting assurance cases:* Although 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 that the evidence is sufficient. Safety arguments typically have some weaknesses, so they cannot be fully trusted on their own. In other words, there is always a question about the level of trust for the safety arguments and cited evidence, which makes a justification for the sufficiency of confidence in safety cases essential. Several attempts have been to quantitatively measure confidence in safety cases, such as in [Bloomfield07, Denney11].

  A new approach for creating clear safety cases was introduced in [Hawkins11] to facilitate their development and increase confidence in the constructed cases. This approach basically separates the major components of safety cases into a safety argument and a confidence argument. A safety argument is limited to arguments and evidence that directly target the system safety—for example, explaining why a specific hazard is sufficiently unlikely to occur and arguing this claim by testing results as evidence. A confidence argument is given separately; it seeks to justify the sufficiency of confidence in the safety argument. For example, questions about the level of confidence in the given testing result evidence (e.g., whether that testing was exhaustive) should be addressed in the confidence argument. These two components, while presented explicitly and separately, are interlinked so that the 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 so as 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 by [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 given as well.

  To show sufficient confidence in a safety argument, the developer of a confidence argument first explores all concerns about the level of 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 when 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, but rather are more commonly inductive. In turn, safety cases are, by their nature, often subjective [Kelly07]. The objective of safety case evaluation, therefore, is to assess whether 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 are common in safety arguments. Therefore, reviewers should be required to express their opinions about only the basic elements in the safety case. A mechanism should then provide a way to aggregate the reviewers' opinions about the basic elements in the safety case so as to communicate a message about its overall sufficiency.

Several approaches to assessing assurance cases have been proposed. The work in [Kelly07] presents a structured approach to assurance case review by focusing primarily on assessment of 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. This framework is based on the notion of eliminative induction—the principle that confidence in the truth of a claim increases as reasons for doubting its truth are identified and eliminated. Defeaters, in contrast, offer possible reasons for doubting. The notion of Baconian probability is then used to provide a measure of confidence in assurance cases based on how many defeaters have been identified and eliminated.

A structured method for assessing the level of sufficiency and insufficiency of safety arguments was outlined in [Ayoub13]. 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 deals with the overall sufficiency of the safety argument. In other words, the approach in [Kelly07] provides a skeleton for a systematic review process; by comparison, 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-Shafer model.

Finally, linguistic scales are introduced in [Cyra08] as a means to express the expert opinions of reviewers and the aggregation results. Linguistic scales are appealing in this context, as they are closer to human nature than are numbers. They are based on qualitative values such as "high," "low," and "very low" and are mapped into the interval for evaluation.

1.3.7.3 Case Study: GPCA Safety

This section builds on the case study of the GPCA infusion pump, which was presented in Section 1.3.3.3. Assurance cases for medical devices have been discussed in [Weinstock09]. The work in [Weinstock09] can be used as starting 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 development approach are likely to lead to similarities in safety arguments. In keeping with this understanding, safety case patterns [Kelly97] have been proposed as means of capturing similarities between arguments. Patterns allow the common argument structure to 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]. In this section, 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. 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 1.8 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

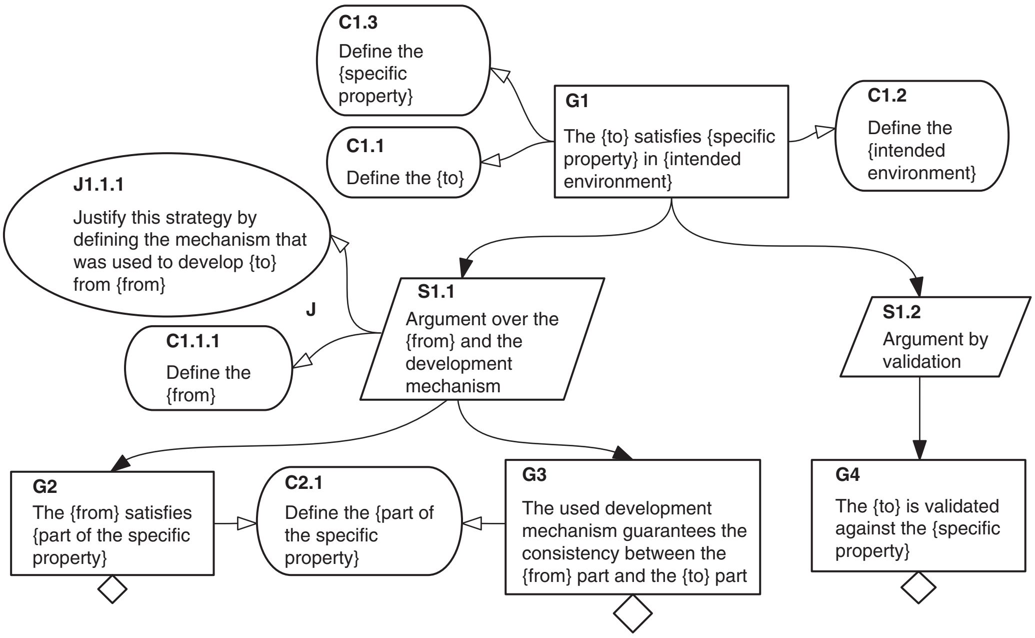

C1.3
Define the {specific property}
G1
The {to} satisfies {specific property} in {intended environment}
C1.2
Define the {intended environment}
C1.1
Define the {to}
J1.1.1
Justify this strategy by defining the mechanism that was used to develop {to} from {from}
J
S1.1
Argument over the {from} and the development mechanism
S1.2
Argument by validation
C1.1.1
Define the {from}
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 1.8: *The proposed from_to pattern*

A. Ayoub, B. Kim, I. Lee, O. Sokolsky. *Proceedings of NASA Formal Methods: 45th International Symposium*, pp. 141–146. With permission from Springer.

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 differing 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).

Figure 1.9 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

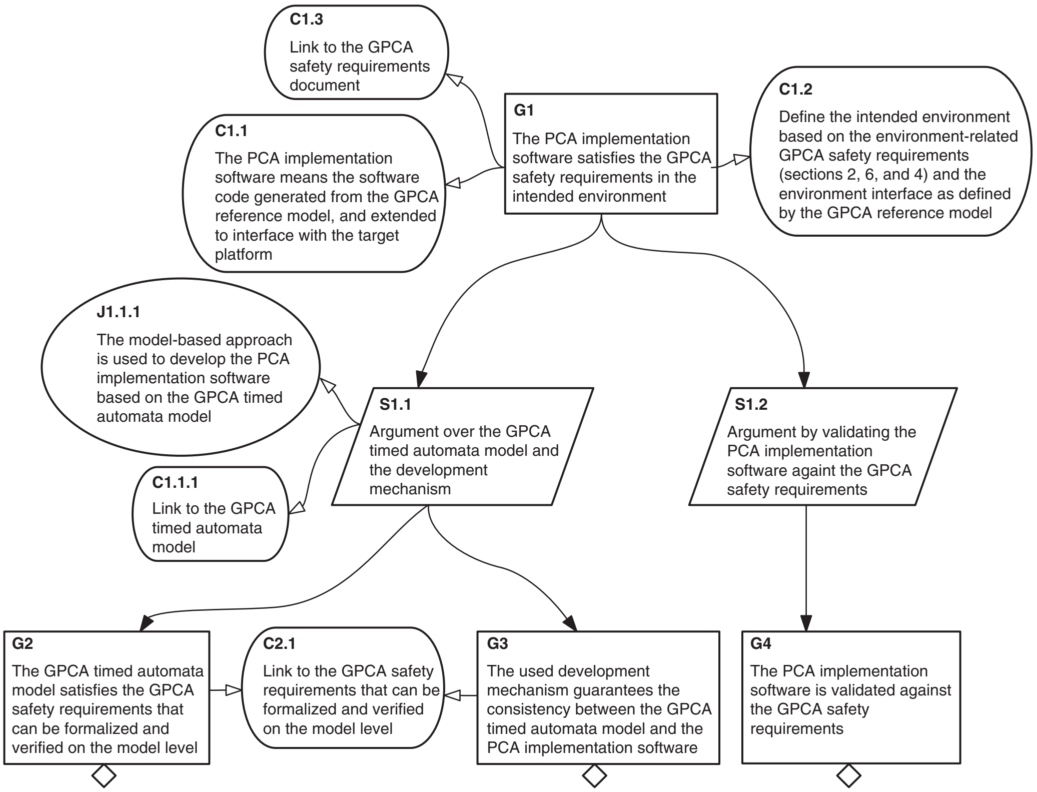

C1.3
Link to the GPCA safety requirements document
G1
The PCA implementation software satisfies the GPCA safety requirements in the intended environment
C1.2
Define the intended environment based on the environment-related GPCA safety requirements (sections 2, 6, and 4) and the environment interface as defined by the GPCA reference model
C1.1
The PCA implementation software means the software code generated from the GPCA reference model, and extended to interface with the target platform
J1.1.1
The model-based approach is used to develop the PCA implementation software based on the GPCA timed automata model
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 1.9: *An instance of the from_to pattern.*

A. Ayoub, B. Kim, I. Lee, O. Sokolsky. *Proceedings of NASA Formal Methods: 45th International Symposium,* pp. 141–146. With permission from Springer.

concerned property. In this case, correct PCA implementation means it satisfies the GPCA safety requirements that were 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) as well as by the consistency between the model and the implementation (G3). The correctness of the GPCA timed automata model (i.e., further development for G2) is proved by applying the UPPAAL model-checker against the GPCA safety requirements, which 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.

Note that not all of 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 or verified against the model given its level of detail (e.g., "The flow rate for the bolus dose shall be programmable" cannot be formalized meaningfully and then verified on the model level).

Note

Generally, using safety case patterns does not necessarily guarantee that the constructed safety case will be sufficiently compelling. Thus, when instantiating the *from_to* pattern, it is necessary to justify 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 Confident As Reasonably Practical) [Hawkins09a].

Assurance Deficit Example

As discussed in Section 1.3.3.3 and shown in Figure 1.3, the GPCA Simulink/Stateflow model was transformed into an equivalent GPCA timed automata model. Although it is relatively straightforward 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 in Figure 1.9) can be stated as "There are semantic differences between the Simulink/Stateflow and the UPPAAL timed automata model." To mitigate this residual assurance deficit, exhaustive conformance testing between the GPCA Simulink/Stateflow model and the GPCA timed automata model may suffice.

1.4 Practitioners' Implications

One can distinguish the following groups of stakeholders in MCPS:

- MCPS developers, including manufacturers of medical devices and integrators of medical information technologies
- MCPS administrators—typically clinical engineers in hospitals, who are tasked with deploying and maintaining MCPS

- MCPS users—clinicians who perform treatment using MCPS
- MCPS subjects—patients
- MCPS regulators, who certify the safety of MCPS or approve their use for clinical purposes

In the United States, the FDA is the regulatory agency charged with assessing the safety and effectiveness of medical devices and approving them for specific uses.

All of the stakeholder 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. In this section we consider each group of stakeholders and identify specific concerns that apply to them as well as unique challenges that they pose.

1.4.1 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, the medical device industry has been plagued with software-related recalls, with 19% of all recalls of medical devices in the United States 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 intense 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 1.3.3, hold the promise of more efficient verification and validation, leading to shorter development cycles.

At the same time, many medical device companies complain about the 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.

1.4.2 MCPS Administrator Perspective

Clinical engineers in hospitals are charged with maintaining the wide variety of medical devices that constitute 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 1.3.4, may help to ensure that more devices are compatible with one another, making the job of maintaining the inventory and the assembly of clinical scenarios easier. This, in turn, reduces treatment errors and improves patient outcomes and, at the same time, saves the hospital money.

1.4.3 MCPS User Perspective

Clinicians use MCPS as part of delivering patient treatments. A specific treatment can, in most cases, be performed with 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 clinical scenarios and virtual medical devices, introduced in Section 1.3.4, can help establish a common user interface for the MCPS, regardless of the specific devices used to implement it. Such an interface would help to reduce clinical errors when using these 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 need 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 health care is the high workload faced by caregivers. Each healthcare provider is likely to be caring for multiple patients and must 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 deliver 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 the patient's condition and reducing the rate of false alarms [Imhoff09].

1.4.4 Patient Perspective

Of all the various stakeholder groups, patients stand to gain the most from the introduction of MCPS. In addition to the expected improvements in the safety of treatments achieved 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 must attend to multiple patients as part of their workload, 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, thereby enabling them to intervene before the change leads to a serious problem. Furthermore, continuous monitoring, combined with support for predictive decision making, similar to the system discussed in Section 1.3.5, will allow treatment to be proactive rather than reactive.

Probably the biggest improvement in the quality of care for patients will come with the transition from general guidelines meant to apply to all patients within a certain population to personalized approaches, in which treatment is customized to the individual needs of the patient and takes into account his or her specific 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.

1.4.5 MCPS Regulatory Perspective

Regulators of the medical devices industry are tasked with assessing the safety and effectiveness of MCPS. The two main concerns that these regulators face are improving the quality of the assessment and making the best use of the limited resources that agencies have available for performing the assessment. These two concerns are not independent, because more efficient ways of performing assessments would allow regulators more time to conduct deeper evaluations. The safety case technologies discussed in Section 1.3.7 may help address both. The move toward evidence-based assessment may allow regulators to perform more accurate and reliable assessments. At the same time, organizing evidence into a coherent argument will help them perform these assessments more efficiently.

1.5 Summary and Open Challenges

This chapter presented a broad overview of trends in MCPS and the design challenges that these trends present. It 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 the safety of a system. Increasingly, model-based development is embraced by the medical devices industry. Even so, the numerous recalls of medical devices that have occurred in recent years demonstrate that the problem of device safety is far from being solved.

The next-level challenge arises from the need to organize individual devices 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. At the same time, 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 the bedside from available devices, creates new regulatory challenges and requires medical application platforms—that is, trusted middleware that can 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.

To fully utilize the promise inherent in multi-device MCPS, new algorithms need to be developed to process and integrate patient data from multiple sensors, provide better decision support for clinicians, produce more accurate and informative alarms, and so on. This need gives rise to two kinds of open challenges. On the one hand, additional clinical research as well as data analysis needs to be performed to determine the best ways of using the new information made available through combining multiple rich data sources. On the other hand, new software tools are needed 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, in which information about the patient's condition, 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 as means to improve glycemic control in patients with diabetes. 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, the high connectivity, and the prevalence of software-enabled functionality make evaluation of such systems' 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://www.adelard.com/asce/choosing-asce/cae.html, 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 *Formal Modeling and Analysis of Timed Systems*. Springer, 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." *Biodevices*, pages 52–60, January 2009.

[ASTM09]. 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." ASTM International, 2009.

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

[Ayoub12]. A. Ayoub, B. Kim, I. Lee, and O. Sokolsky. "A Safety Case Pattern for Model-Based Development Approach." In *NASA Formal Methods*, pages 223–243. Springer, 2012.

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

[Becker09]. U. Becker. "Model-Based Development of Medical Devices." *Proceedings of the Workshop on Computer Safety, Reliability, and Security (SAFECERT)*, Lecture Notes in Computer Science, vol. 5775, 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*, Lecture Notes in Computer Science, pages 200–237. Springer, 2004.

[Bequette03]. B. Bequette. *Process Control: Modeling, Design, and Simulation*. Prentice Hall, 2003.

[Bloomfield07]. R. Bloomfield, B. Littlewood, and D. Wright. "Confidence: Its Role in Dependability Cases for Risk Assessment." 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 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*, vol. 27, no. 2–3, pages 137–146, 2003.

[Clarke07]. M. Clarke, D. Bogia, K. Hassing, L. Steubesand, T. Chan, and D. Ayyagari. "Developing a Standard for Personal Health Devices Based on 11073." 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 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 and Engineering Sciences*, vol. 367, pages 411–429, 2009.

[Clinical07]. Clinical Alarms Task Force. "Impact of Clinical Alarms on Patient Safety." *Journal of Clinical Engineering*, vol. 32, no. 1, pages 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." *IEEE Reviews in Biomedical Engineering*, vol. 2, 2009.

[Commission13]. The Joint Commission. "Medical Device Alarm Safety in Hospitals." *Sentinel Event Alert*, no. 50, April 2013.

[Cyra08]. L. Cyra and J. Górski. "Expert Assessment of Arguments: A Method and Its Experimental Evaluation." International Conference on Computer Safety, Reliability and Security (SAFECOMP), 2008.

[Denney11]. E. Denney, G. Pai, and I. Habli. "Towards Measurement of Confidence in Safety Cases." International Symposium on Empirical Software Engineering and Measurement (ESEM), Washington, DC, 2011.

[Dias07]. A. C. Dias Neto, R. Subramanyan, M. Vieira, and G. H. Travassos. "A Survey on Model-Based Testing Approaches: A Systematic Review." *Proceedings of the 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. Shvo. "HL7 Clinical Document Architecture, Release 2." *Journal of the American Medical Informatics Association*, vol. 13, no. 1, pages 30–39, 2006.

[Donchin02]. Y. Donchin and F. J. Seagull. "The Hostile Environment of the Intensive Care Unit." *Current Opinion in Critical Care*, vol. 8, pages 316–320, 2002.

[Edworthy06]. J. Edworthy and E. Hellier. "Alarms and Human Behaviour: Implications for Medical Alarms." *British Journal of Anaesthesia*, vol. 97, pages 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*, vol. 268, pages 2420–2425, 1992.

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

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

[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*, vol. 293, pages 1223–1238, 2005.

[Ginsberg09]. B. H. Ginsberg. "Factors Affecting Blood Glucose Monitoring: Sources of Errors in Measurement." *Journal of Diabetes Science and Technology*, vol. 3, no. 4, pages 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*, vol. 39, no. 3, pages 194–199, 2005.

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

[Harris13]. Harris Healthcare (formerly careFX). www.harris.com.

[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." *Proceedings of the IEEE/ACM 3rd International Conference on Cyber-Physical Systems (ICCPS)*, pages 3–12, Washington, DC, 2012.

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

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

[Hawkins11]. R. Hawkins, T. Kelly, J. Knight, and P. Graydon. "A New Approach to Creating Clear Safety Arguments." In *Advances in Systems Safety*, pages 3–23. Springer, 2011.

[Henzinger07]. T. A. Henzinger and C. M. Kirsch. "The Embedded Machine: Predictable, Portable Real-Time Code." *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 29, no. 6, page 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*, vol. 25, no. 4, page 905, 2004.

[Imhoff06]. M. Imhoff and S. Kuhls. "Alarm Algorithms in Critical Care Monitoring." *Anesthesia and Analgesia*, vol. 102, no. 5, pages 1525–1536, 2006.

[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*, vol. 23, no. 1, pages 39–50, 2009.

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

[ISO/IEEE11073]. ISO/IEEE 11073 Committee. “Health Informatics—Point-of-Care Medical Device Communication Part 10103: Nomenclature—Implantable Device, Cardiac.” 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?* Committee on Certifiably Dependable Software Systems, National Research Council. National Academies Press, May 2007.

[Jee10]. E. Jee, I. Lee, and O. Sokolsky. “Assurance Cases in Model-Driven Development of the Pacemaker Software.” 4th International Conference on Leveraging Applications of Formal Methods, Verification, and Validation, Volume 6416, Part II, ISoLA’10, pages 343–356. Springer-Verlag, 2010.

[Jeroeno4]. J. Levert and J. C. H. Hoorntje. “Runaway Pacemaker Due to Software-Based Programming Error.” *Pacing and Clinical Electrophysiology*, vol. 27, no. 12, pages 1689–1690, December 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.” Workshop on Assurance Cases for Security: The Metrics Challenge, Dependable Systems and Networks (DSN), 2007.

[Kelly97]. T. Kelly and J. McDermid. “Safety Case Construction and Reuse Using Patterns.” International Conference on Computer Safety, Reliability and Security (SAFECOMP), pages 55–96. Springer-Verlag, 1997.

[Kelly04]. T. Kelly and R. Weaver. “The Goal Structuring Notation: A Safety Argument Notation.” DSN 2004 Workshop on Assurance Cases, 2004.

[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." *Embedded Software (EMSOFT)*, pages 155–164, Taipei, Taiwan, 2011.

[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." 23rd IEEE International Symposium on Rapid System Prototyping (RSP), pages 16–22, 2012.

[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." *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 Technology Society*, vol. 3, no. 1, pages 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*, vol. 39, no. 4, pages 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*, vol. 100, no. 1, pages 75–90, January 2012.

[Lofsky04]. A. S. Lofsky. "Turn Your Alarms On." *APSF Newsletter*, vol. 19, no. 4, page 43, 2004.

[Lublinerman09]. R. Lublinerman, C. Szegedy, and S. Tripakis. "Modular Code Generation from Synchronous Block Diagrams: Modularity vs. Code Size." *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2009)*, pages 78–89, New York, NY, 2009.

[Lynn11]. L. A. Lynn and J. P. Curry. "Patterns of Unexpected In-Hospital Deaths: A Root Cause Analysis." *Patient Safety in Surgery*, vol. 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, August 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 Prototype in PVS." *Proceedings of the 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, 2013.

[Mazoit07]. J. X. Mazoit, K. Butscher, and K. Samii. "Morphine in Postoperative Patients: Pharmacokinetics and Pharmacodynamics of Metabolites." *Anesthesia and Analgesia*, vol. 105, no. 1, pages 70–78, 2007.

[McMaster13]. Software Quality Research Laboratory, McMaster University. Pacemaker Formal Methods Challenge. http://sqrl.mcmaster.ca/pacemaker.htm.

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

[MDPNP]. MD PnP: Medical Device "Plug-and-Play" Interoperability Program. 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, 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*, vol. 23 (Supplement 1), pages 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*, vol. 15, pages 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." *IEEE Transactions on Industrial Informatics*, PP(99):1–1, 2012.

[Phillips10]. Phillips eICU Program. http://www.usa.philips.com/healthcare/solutions/patient-monitoring.

[Rae03]. A. Rae, P. Ramanan, D. Jackson, J. Flanz, and D. Leyman. "Critical Feature Analysis of a Radiotherapy Machine." International Conference of Computer Safety, Reliability and Security (SAFECOMP), September 2003.

[Sapirstein09]. A. Sapirstein, N. Lone, A. Latif, J. Fackler, and P. J. Pronovost. "Tele ICU: Paradox or Panacea?" *Best Practice and*

Research Clinical Anaesthesiology, vol. 23, no. 1, pages 115–126, March 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*, vol. 67, pages 1207–1224, 1979.

[Simone13]. L. K. Simone. "Software Related Recalls: An Analysis of Records." *Biomedical Instrumentation and Technology*, 2013.

[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.

[Wagner10]. S. Wagner, B. Schatz, S. Puchner, and P. Kock. "A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models." *International Symposium on Software Reliability Engineering*, 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 Devices." Technical Report, CMU/SEI-2009-TN-018, 2009.

[Ye05]. F. Ye and T. Kelly. "Contract-Based Justification for COTS Component within Safety-Critical Applications." PhD thesis, Department of Computer Science, University of York, 2005.

Index