Model-Driven Safety Analysis of Closed-Loop Medical Systems

Miroslav Pajic, Rahul Mangharam, Oleg Sokolsky, David Arney, Julian Goldman and Insup Lee

Abstract— In modern hospitals, patients are treated using a wide array of medical devices that are increasingly interacting with each other over the network, thus offering a perfect example of a cyber-physical system. We study the safety of a medical device system for the physiologic closed-loop control of drug infusion. The main contribution of the paper is the verification approach for the safety properties of closed-loop medical device systems. We demonstrate, using a case study, that the approach can be applied to a system of clinical importance. Our method combines simulation-based analysis of a detailed model of the system that contains continuous patient dynamics with model checking of a more abstract timed automata model. We show that the relationship between the two models preserves the crucial aspect of the timing behavior that ensures the conservativeness of the safety analysis. We also describe system design that can provide open-loop safety under network failure.

Index Terms—closed-loop medical systems, model-based development, safety analysis

I. INTRODUCTION

Clinical scenarios for critical care patients often involve large numbers of medical devices. Some of these devices, such as bedside monitors, provide vital information about the state of the patient. Other devices, for example, infusion pumps, provide treatment. That is, they affect the state of the patient, for example, by infusing medication. Medical device systems, considered together with the patient and caregivers, represent an important class of cyber-physical systems. Patient safety is the primary concern in such systems, yet reasoning about patient safety is very difficult because of insufficient understanding of the dynamics of human body response to treatment. Human errors, another important source of patient safety problems, are also difficult to reason about in the framework of conventional embedded system development.

It is natural to view a clinical scenario as a control system, in which the patient is a plant, bedside monitors are sensors and infusion pumps are actuators. Traditionally, caregivers perform the role of the controller in such a system. This means that the caregiver needs to continuously monitor all sensor devices and apply appropriate treatment. The large number of devices to monitor and control makes the job of the caregiver very difficult. On top of that, a caregiver is typically responsible for several patients. An emergency may divert the caregiver's attention elsewhere, making him or her miss an important event. As a result, patient safety may suffer. Multiple such occurrences are documented in the clinical literature.

The human caregiver will always play an indispensable part of most clinical scenarios. However, in many cases automatic controllers can play important roles, reducing the burden on the caregiver and avoiding the possibility of human errors. A number of such cases has been documented in the ASTM standard for the Integrated Clinical Environment [1], developed by the Medical Device Plug-and-Play Interoperability program(mdpnp.org) [2]. Although many medical devices today have network interfaces and can send sensed data across the network, few can be controlled remotely. Vendors of medical equipment continue to stay away from closed-loop scenarios.

The rationale is that it is difficult to reason about patient safety in closed-loop scenarios. Such reasoning is necessary for medical devices, which have to be approved for use by government regulators who assess their safety and effectiveness. A particular challenge arises from the complexity of a complex interplay between the continuous dynamics of the patient reaction to treatment and the discrete nature of the controller and communication network. The dynamics of the patient body is not well understood and exhibits parametric uncertainty and high variability between different patients.

To overcome this difficulty, we explore a model-driven approach that allows us to prove safety properties of devices on the modeling level and ensures that abstract models used in the verification process are sound with respect to the actual dynamics of the system. Both the abstract, formal model and the detailed, informal model are needed in the process of verification, validation, and regulatory approval of closed-loop medical device systems. On the one hand, formal models allow us to exhaustively explore the possible behaviors of the system and prove its safety. On the other hand, detailed models allow us to use high-fidelity simulation that take real system dynamics into account. Both kinds of results can be used to make the case for regulatory approval.

We start with a detailed model of the system in Simulink. The model contains a patient model that reflects continuous pharmacokinetic dynamics of drug absorption by the patient body. To reflect the variation between different patient populations, the patient model incorporates uncertainty in the values of its parameters. The Simulink model enables high-fidelity simulations of the system in various scenarios. Yet, currently there are no tools supporting the verification of hybrid systems (i.e., systems consisted of both discrete-time components and continuous-time dynamics) modeled in Simulink that are both nondeterministic and with uncertain parameters. Most existing tools for Simulink are limited to testing-based analysis that does not guarantee the coverage of parameter space.

M. Pajic and R. Mangharam are with the Department of Electrical & System Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA USA e-mail: {pajic, rahulm}@seas.upenn.edu.

D. Arney, I. Lee and O. Sokolsky are with the Department Computer & Information Science, University of Pennsylvania, Philadelphia, PA 19104, USA USA e-mail: {sokolsky, arney,lee}@seas.upenn.edu.

J. Goldman is with Mass. General Hospital & CIMIT, Boston, MA 19104, USA USA e-mail: jmgoldman@partners.org.

To prove safety of the system, we express the safety property as a timing relationship between system components. We then model the system using the formalism of *timed automata* in the UPPAAL tool [3].[1] The model abstracts away continuous dynamics of the system, replacing it with timing constraints. The values for the timing constraints are obtained from the continuous dynamics of system components using the detailed Simulink model. It is necessary to guarantee that the patient model in UPPAAL over-approximates the patient's dynamics with respect to the utilized control algorithm. For that, we present a procedure that utilizes convex optimization to calculate an upper bound on critical timing values for a linear time-invariant (LTI) model with uncertain parameters and show that behaviors of the UPPAAL model is always within this bound.

There is a need to develop a methodology for the analysis of safety properties of closed-loop medical device systems [4]. In this paper we present a case-study focused on a system of clinical importance. In the case study, we prove that the system is safe under a set of assumptions. We then demonstrate that a violation of these assumptions (e.g., by a more realistic fault model) can make the system unsafe. Finally, we propose a solution to restore system safety under the new fault model. The approach presented in this paper is intended to serve as a reference point, a step toward a model-based methodology for safety analysis of closed-loop medical device systems.

The rest of the paper is organized as follows. Section II introduces the medical case study and presents the system architecture that we used to describe our approach. Section III explains the use of Simulink models of the system to analyze safety with no failure assumption. It also describes a procedure used to translate safety requirements into timing constraints in cases where system is modeled as a continuous-time LTI system with uncertain parameters. Section IV describes the UPPAAL models of the system and safety properties we have verified. Section V proposes modifications to the system to deal with failures and provides an argument that the new system guarantees open-loop safety. The last section summarizes the paper and identifies the future work.

II. CASE STUDY

In this section, we describe a case study that represents one of the MD PnP interoperability clinical scenarios [1]. In the past, we have built a demonstration of this scenario [5]. Several variants of this system have been presented at the American Society of Anesthesiologists annual meeting in 2007, where it won first place in the scientific exhibits, at the 2008 HIMMS (Healthcare Information and Management Systems Society) Congress, and at the 2008 CIMIT Innovation Congress.

We have built the system according to the Integrated Clinical Environment (ICE) architecture, developed by the MD PnP project. Figure 1 shows the main components of an ICE-compliant system. The patient and caregiver are the human elements of the system. The Supervisor is the computer system that runs the control algorithm. Medical devices are connected, through adapters where necessary, to the Network Controller, which keeps track of connected devices and their capabilities. The Data Logger records pertinent network traffic
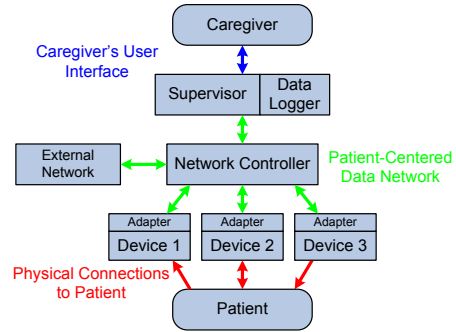

Fig. 1. MD PnP Architecture with Patient-in-the-Loop Automatic Control.

for later forensic analysis and external networks such as the hospital information system are connected through an external interface. In our case study, we did not model the data logger since it does not affect our runtime safety analysis.

A. Clinical Use Case

The selected scenario involves a patient connected to a patient-controlled analgesia (PCA) infusion pump. PCA infusion pumps are commonly used to deliver opioids for pain management, for instance after surgery. Patients have very different reactions to the medications and require very different dosages and delivery schedules. PCA pumps give the patient a button to press to request a dose when they decide they want it rather than using a schedule fixed by a caregiver. Some patients may decide they prefer a higher level of pain to the nausea the drugs may cause and can press the button less often, while patients who need a higher dose can press it more often.

A major problem with opioid medications in general is that an excessive dose, or overdose, can cause respiratory failure. A properly programmed PCA system should not allow an overdose because it is programmed with limits on the number of doses it will deliver, regardless of how often the button is pushed. However, this safety mechanism is not sufficient to protect all patients. Some patients still receive overdoses if the pump is misprogrammed, if the pump programmer overestimates the maximum dose a patient can receive, if the wrong concentration of drug is loaded into the pump, or if someone other than the patient presses the button (PCA-by-proxy), among other causes. PCA pumps have been involved in a large number of adverse events [6], and existing safeguards such as drug libraries and programmable limits are not adequate to address all the scenarios seen in clinical practice [7].

The system we are considering aims to improve patient safety in such scenarios by introducing a supervisor that monitors patient data for the early signs of respiratory failure and can stop the infusion and sound an alarm if the patient experiences an adverse event. We use a pulse oximeter device that receives physiological signals from a clip on the patient's finger and processes them to calculate heart rate and SpO_2 outputs, where SpO_2 is the measure of blood oxygenation. Note that, at the time of writing, there are no PCA pumps on the market that are capable of being remotely controlled. In the demonstration system, we used the PCA pump prototype we have built during the Generic Infusion Pump project [8].

---

[1]Due to space constraints, we do not present the formal definition and semantics of timed automata. These definitions can be obtained from [3].
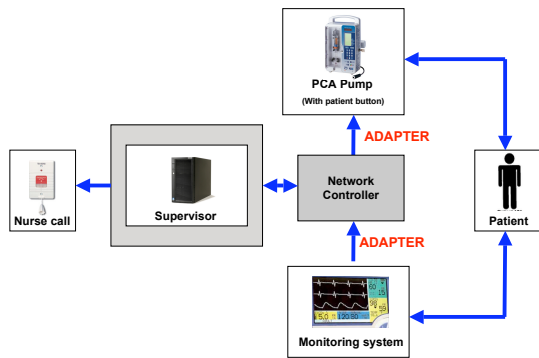
Fig. 2.   Hardware for PCA Demo System.

B. System Architecture

Figure 2 shows the components of the PCA safety system. The system components are described below. Figure 3 shows the devices and essential data flow in this control loop. In the case-study pulse oximeter acts as the Monitoring System from Figure 2. It receives physiological signals from the patient and processes them to produce heart rate and SpO_2 outputs. The Supervisor gets these outputs and makes a control decision, possibly sending a stop signal to the PCA Pump. Unless it is stopped by the Supervisor, the PCA pump delivers a drug to the patient at its programmed rate that corresponds to the selected operating mode. The patient model gets the drug rate as an input and updates the drug level in the patient's body. This in turn influences the physiological output signals through a drug absorption function.

1) PCA Infusion Pump: The PCA pump in the case study operates in the following way. Before operation, the pump is programmed by the caregiver, who sets the pump's operating mode, normal rate of infusion, the increased rate of a bolus, and bolus duration. Some PCA pumps also can be programmed to limit the total amount of drug to be infused. Once programmed and started, the pump delivers the drug at the normal rate until it is stopped or the bolus button is pressed. From that moment, it delivers drug at the bolus rate for the specified duration and then returns to the normal rate.

The pump is equipped with a number of built-in sensors that detect internal malfunctions such as the presence of air in the tubes that deliver the drug. When a problem is detected, the pump is stopped. We do not consider such malfunctions in this case study and do not represent the built-in alarm mechanism.

Finally, the pump is equipped with a network interface, which allows the pump to transmit its status across the network to other devices such as the logger. For the purpose of our scenario, we assume that the network interface allows the pump to accept control signals. A *stop* control signal will set the current infusion rate to zero, while the *start* signal will set the normal infusion rate (regardless of the state of the pump before it was stopped).

2) Monitoring System: Patients using a PCA pump are usually also attached to patient monitors that record the patient's EKG, blood pressure, respiratory rate, and SpO_2. These monitors sound alarms if the measured values are outside thresholds set by the caregivers, but do not stop the infusion. Thus, the patient continues to receive more of an overdose while the caregiver responds, assesses the patient, and if a real problem occurred, finally stops the pump.

last output value	97 - 100	94 - 96	90 - 93	85 - 89	< 85
new window size	10	8	7	6	4

TABLE I
SLIDING WINDOW SIZE FOR PULSE OXIMETER

In this study, we look at using SpO_2 and heart rate measurements as the basis for a physiologic closed-loop control system that can stop the PCA pump and halt the dose of opioid while sounding an alarm if respiratory distress is detected. Both of these measurements can be produced by a device called *Pulse Oximeter*. This device is equipped with a finger clip sensor that shines two wavelengths of light through the patient's finger. The measured light intensity reflects the blood oxygen content, which can change rapidly.

The pulse oximeter samples the patient's SpO_2 at regular intervals, processes them, and outputs an averaged result [9]. It calculates the average using a variable-sized sliding window. The window size varies with the last output value. The reason for changing the window size is that a smaller sample size gives faster, but potentially less accurate results. When SpO_2 values are low, quick response is more important than filtering out transient noise. When SpO_2 is high, increasing the window size helps to filter out transient low values at the expense of less frequent updates. Since the samples are at regular intervals and a varying number of samples are used to calculate the output, the output is updated irregularly. The size of the sliding window used in the case study is determined using a simple table shown in Table I. Note that this table does not reflect the details of any real implementation but rather attempts to capture the essential behavior of a typical pulse oximeter.

3) Caregiver Model: The caregiver in this system programs the PCA pump and reacts to alarms. The control system is closed loop, so no intervention by the caregiver is necessary to stop the infusion when a problem is detected. The caregiver can react to restart the system if it has stopped in reaction to a false alarm, or when a problem such as a slipped patient sensor is fixed.

4) Patient Model: To model the Patient we use a pharmacokinetic patient model for intravenous delivery of anesthetic drugs presented in [10] (module 12). The model utilizes a common 3-compartment model to describe the changes in drug concentration in blood and tissue. The patient is described as a continuous-time LTI system with a state space representation:
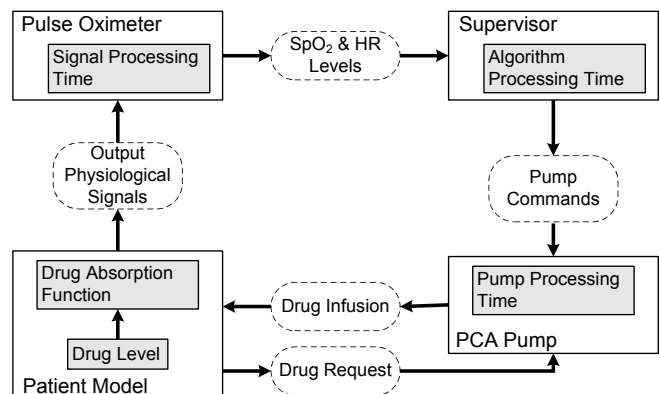

Fig. 3.   PCA System Control Loop.

$$\begin{bmatrix} \dot{C_1} \\ \dot{C_2} \\ \dot{C_3} \end{bmatrix} = \underbrace{\begin{bmatrix} -(k_{12}+k_{13}+k_{10}) & k_{21} & k_{31} \\ k_{12} & -k_{12} & 0 \\ k_{13} & 0 & -k_{31} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{1}{V_1} \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{B}} I$$

$$dl = \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{\mathbf{C}} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} \qquad (1)$$

where C_i, $i \in \{1,2,3\}$ is the drug concentration in the ith compartment, I is the mass infusion rate (in mass/unit of time) of the drug, k_{ij} are patient specific constants and V_1 is the volume of the blood plasma compartment. Since compartment 1 presents blood plasma, its concentration is mapped into the drug level concentration dl as in (1).

Using the above model, we represent the instantaneous level of medication in the patient's body as a single variable dl. This variable is linked to the patient's heart rate and SpO_2 by the drug absorption function, which represents how the patient reacts to the dose received over time. Using dl value, the state space of the patient is partitioned into three regions: in pain (under-medicated), pain-controlled (adequate medication), or over-medicated. These regions, shown in Figure 4, are defined as follows:

• The *Safe* region is defined as the region where the *Patient*'s readings are below some predefined threshold values that guarantee that *Patient*'s vitals are not endangered.
• If the patient is over-medicated to the point that he or she starts experiencing respiratory distress, we consider it an overdose. Thus, we define the *Critical* region where *Patient*'s life is in danger or there is a chance that irreparable damage can occur. The overdose condition is referred as border of the *Critical* region.
• The *Alarming* region defined as the region where *Patient*'s vitals are not endangered but there is a reasonable concern that the *Patient* can be forced to the *Critical* region.

Any treatment needs to make sure that the patient stays out of the critical region, and we use this requirement as the main safety property of the system that needs to be ensured. In this case study, since the drug level is estimated from the patient's SpO_2 and heart rate, we defined the boundary of the *Critical* region in terms of these values and set it to $H_2^{SpO_2} = 70\%$ for SpO_2 (and $H_2^{HR} = 11.5\ beats/min$ for heart rate), a clear indication of respiratory failure.

It is worth noting here that some patients react very quickly to a drug dose, while others react more slowly. Furthermore, a mapping between the drug level on one side and measured SpO_2 level and heart rate is also patient specific. Therefore, by adjusting this function, we can tune the model to different patient types. However, in all these cases patient reaction to drug can be described as in (1), where patient specific
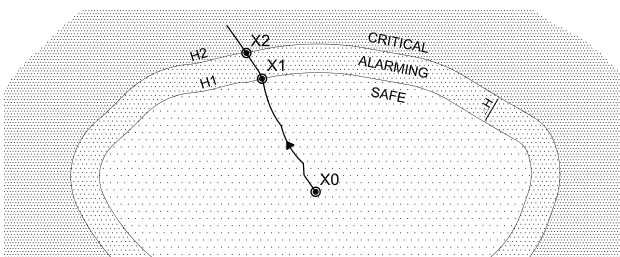

Fig. 4. Regions of *Patient*'s conditions.

coefficients belong to predefined regions and fully determine the patient's behavior.

To analyze the dynamics of the patient's pharmacokinetic model, we have designed the system model in Simulink (the model is described in next section). This enables simulation and analysis of the system's behavior under different scenarios.

5) Supervisor: The supervisor is a running program in the system that communicates with other devices and executes clinical application scripts (CAS's). A CAS is a script that implements a particular clinical use case. The clinical application in this case study is to control the loop shown in Figure 3. The Supervisor receives the patient's heart rate and SpO_2 measurements from the pulse oximeter and uses this information to decide whether the PCA infusion pump should be allowed to run or immediately stopped.

In the case study, two simple Supervisor's control algorithms were designed. In the first algorithm, after the pump is activated the decision to stop the pump is made as soon as the patient heart rate or SpO_2 readings fall below a fixed threshold. The threshold value needs to ensure that the patient does not enter the *Critical* region despite the delay in detecting the problem and delivering the control signal to the pump. For the case study, we defined the threshold as $H_1^{SpO_2} = 90\%$ for the SpO_2 and $H_1^{HR} = 57\ beats/min$ for heart rate. Values below these thresholds typically indicate "a clinical concern" ([11], p. 45), meaning that a caregiver needs to be notified. The supervisor notifies the caregiver when the threshold is crossed, as it sends the message to stop the pump. Values between H_1 and H_2 are thus referred as the *Alarming* region. The width of the alarming region is denoted as $\Delta H = \mid H_2 - H_1 \mid$.

The second control algorithm is similar, with a small difference that in this case the pump is always activated for a fixed, predefined duration of time. Therefore, the Supervisor does not need to send commands to stop the pump unless, as in the first design, it detects that the Patient have entered the *Alarming* region. In this case, the Supervisor again sends a command to stop the pump and alerts the caregiver.

III. MODELING SYSTEM'S DYNAMICS IN SIMULINK

The overall structure of the Simulink model follows that of the model shown in Figure 2. The Simulink blocks are used to capture the dynamics of the PCA infusion pump, Pulse Oximeter (PO), Patient model, and Supervisor, described in Section II. In addition, they are used to define the notion of the safe, critical, and alarming regions precisely. Using the overall structures and timing properties of the components, this section describes when the system is safe and when it is not safe. Finally, it identifies the limitations of the model with respect to its use in system verification.

A. Simulink Models

The PO is implemented as described in Section II (a more detailed description of the PO design is presented in Section IV). It monitors the *Patient*'s HR and SpO_2 level and informs the Supervisor about these values. Figure 5 presents the Patient's model, where effects of the drug flow are depicted using the Patient model described in Section II. Therefore, the Patient is modeled as a continuous-time LTI system where matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are defined as in (1). For the Patient
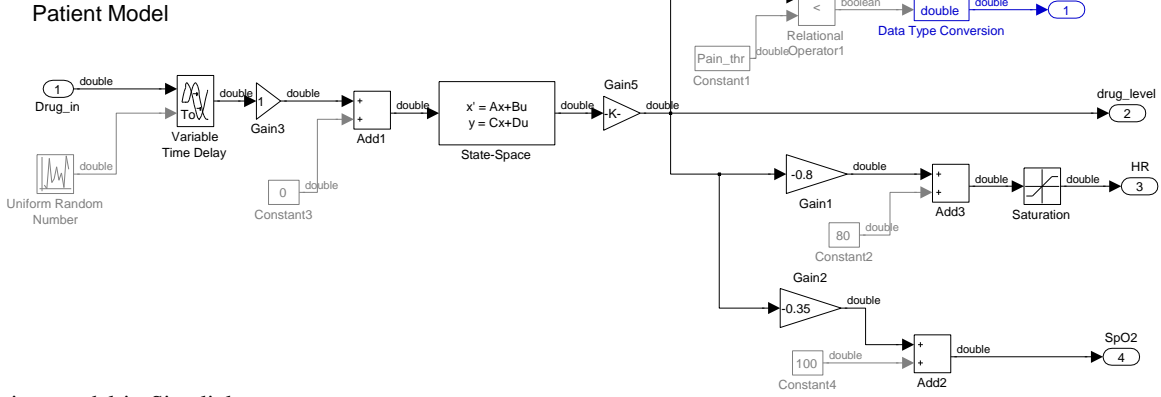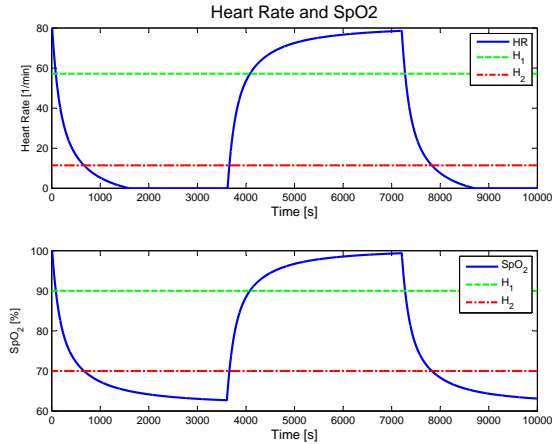
Fig. 5.   Patient model in Simulink.

Fig. 6.   *Patient*'s response on the pump activities.

model that describes a particular population of subjects, we use the parameters for a scenario of intravenous delivery of anesthetic drugs ([10], page 694). Here, the kinetic values in (1) are initialized as:

$$\hat{k}_{10} = 0.152 \text{ min}^{-1}, \hat{k}_{12} = 0.207 \text{ min}^{-1}, \hat{k}_{13} = 0.040 \text{ min}^{-1}$$
$$\hat{k}_{21} = 0.092 \text{ min}^{-1}, \hat{k}_{31} = 0.048 \text{ min}^{-1}, \hat{V}_1 = 12 \text{ liters.} \quad (2)$$

In the general case, to be able to express Patient dynamics for different types of patient populations, it is not possible to fix a value for each of the parameters k_{ij}. Therefore, to model the Patient's dynamics in the general case, each parameter k_{ij} is initialized with a value within a closed region:

$$k_{ij} \in \left[\hat{k}_{ij} - \Delta k_{ij}, \hat{k}_{ij} + \Delta k_{ij} \right] \quad (3)$$
$$V_1 \in \left[\hat{V}_1 - \Delta V, \hat{V}_1 + \Delta V \right],$$

which is a simplified version of the function presented in [12]. This effectively means that matrices $\mathbf{A}$ and $\mathbf{B}$, defined in (1), in the general case (i.e., for all types of patients) belong to a specific polyhedron. The patient's Heart Rate (HR) and SpO_2 level are extracted from its drug level using a linear mapping, as shown in Figure 5. As the mapping between the drug level, measured SpO_2 level and heart rate is patient-specific, it is necessary to include this additional level of uncertainty into the model. Therefore, in the implemented model, the gain in matrix $\mathbf{C}$ (i.e., element C_{11}) can also take values within a pre-specified region $[1 - \Delta c, 1 + \Delta c]$.

To analyze the patient's dynamics for parameter values from (2), the patient's behavior was simulated for a case when

the drug is repeatedly delivered for 1 hour followed with a 1-hour pause. Figure 6 presents the obtained changes in HR and SpO_2 levels during the period of almost 3 hours.

B. Analysis of System Safety Properties

For the aforementioned system we consider the safety requirement that the PCA pump will always be stopped before Patient's *Critical* region is reached. Since the Supervisor can be configured to operate in two different modes, depending on the Supervisor's mode the safety requirement could be translated into different conditions imposed on the system. Therefore, in this section the safety requirement is analyzed separately for both Supervisor modes.

1) Safety conditions for the first Supervisor design: In the first supervisor's design, patient's requests automatically activate the drug flow. The Supervisor is implemented to stop the PCA pump only if the Patient reaches the *Alarming* region. Otherwise, while the Patient is in the *Safe* region, the pump provides continuous drug flow to the patient. We note that the safety requirement (that the Patient never enters into the *Critical* region) is satisfied if the following condition is met:

$$t_{POdel} + t_{net} + t_{Sup} + t_{net} + t_{Pump} + t_{P2PO} + t_{PCA2P} \leq t_{crit} \quad (4)$$

where:

- t_{POdel} - worst case delay caused by PO; it can be calculated from the PO specification, see Table I,
- t_{net} - worst case network delay; the value depends on used network protocol,
- t_{Sup} - worst case delay introduced by the *Supervisor*; it can be calculated from the *Supervisor* model,
- t_{Pump} - worst case delay introduced by the *PCA pump*; the value can be calculated from the pump model,
- t_{P2PO} - worst case latency from the moment when command is sent from the *PCA pump* until the drug starts (or stops) flowing; it depends on type of connection used,
- t_{PCA2P} - worst case delay between the moment the drug flow is activated and the moment when it reaches the Patient; it also depends on the type of used connection,
- t_{crit} - *Patient*'s critical time, a shortest time that *Patient* spends in *Alarming* region before it enters *Critical* region; it can be calculated from the *Patient* model (see below).

If (4) is satisfied, it can be guaranteed that the Supervisor will be able to determine that the Patient has entered the *Alarming* zone and stops the PCA pump before the Patient

switches from the *Alarming* to the *Critical* zone. The values for the time constants are shown in Table II. All values, beside t_{net}, can be obtained from the system specification. Here, for the safety analysis we assume that the network latency t_{net} is bounded by $0.5s$ and we will try to relax the bound later.

Remark 1: Currently there exist several Simulink-based network simulators that can be used for the network modeling (e.g., TrueTime). However, from the perspective of safety analysis and relation (4), the network can be modeled as a random delay with delay distribution that matches the desired application scenario. It is worth noting here that the network induced delay is not fixed and it depends on the underlying network controller (e.g., 802.11 MAC with distributed (DCF) or point (PCF) coordination function) and environmental conditions (e.g., traffic, presence of interference). Furthermore, although implemented network protocols can improve the overall Quality of Service (QoS), in the general case when wireless networks are used the delay can not be bounded, no matter which communication protocol is utilized. Therefore, even though we start with the assumption on bounded network delay, we also consider the cases with unbounded network delay (i.e., messages can be lost), along with its consequences on the system's safety. □

The main problem in determining whether the safety condition is met is to find the value for t_{crit}. If a mathematical model for the patient is known, and patient-specific (fixed) parameters can be obtained, this value can be analytically or numerically determined. In our model, for a specific population of subjects we can use fixed parameters as in (2). Since the Patient is modeled as an LTI process when the drug flow is on, time-responses for drug level, HR and SpO$_2$ level have the same general form:

$$c_{min} + a_1 e^{-\lambda_1 t} + a_2 e^{-\lambda_2 t} + a_3 e^{-\lambda_e t} \qquad (5)$$

where $\lambda_1 = -0.0079$, $\lambda_2 = -0.0023$, $\lambda_3 = -0.0007$ are eigenvalues of matrix $\mathbf{A} = \hat{\mathbf{A}}$ (from (2)) and constants c_{min}, a_1, a_2 and a_3 belong to different sets of constants (depending whether we consider the drug level, HR or SpO$_2$) for a fixed drug infusion rate.

To calculate t_{crit} consider time instances, t_1 and t_2, when the Patient enters the *Alarming* and *Critical* regions, respectively. If the Patient is continuously pushed toward the *Critical* region (which in this case means that the pump is continuously delivering drug to the Patient), then $t_{crit} = t_2 - t_1$. Considering the *Alarming* region boundary for SpO$_2$ it holds: $H_i^{\text{SpO}_2} = c_{min}^{\text{SpO}_2} + a_1^{\text{SpO}_2} e^{-\lambda_1 t_i} + a_2^{\text{SpO}_2} e^{-\lambda_2 t_i} + a_3^{\text{SpO}_2} e^{-\lambda_e t_i}$. Thus, we can calculate t_{crit} from $H_1^{\text{SpO}_2} - H_2^{\text{SpO}_2} = \Delta H^{\text{SpO}_2}$.

For our model the Patient's dynamics is mostly determined by the maximal eigenvalue λ_3, which is approximately $0.0007 s^{-1}$, but the other eigenvalues have effect too. Thus, t_{crit} is a couple of orders of magnitude bigger than the sum of all other timing parameters from (4). For $H_1^{\text{SpO}_2} = 90\%$ and $H_2^{\text{SpO}_2} = 70\%$ we have $t_{crit} \approx 581.2 \ s \approx 9.68$ min. Therefore, for these patient parameters that correspond to a particular population, our system always satisfies the safety requirement if the assumption that all messages are delivered is valid.

time delay	t_{POdel}	t_{net}	t_{Sup}	t_{Pump}	t_{P2PO}	t_{PCA2P}
value	$1s$	$0.5s$	$0.2s$	$0.1s$	$2s$	$2s$

TABLE II
WORST CASE DELAYS

2) Safety Analysis for the System with Uncertain Parameters: The aforementioned procedure for t_{crit} calculation can be used only when exact values for parameters used in matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are known. As mentioned earlier in the section, in the general case we can only claim that matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ belong to polyhedrons $\Re\{\mathbf{A}\}$, $\Re\{\mathbf{B}\}$ and $\Re\{\mathbf{C}\}$, respectively. For example, a polyhedron $\Re\{\mathbf{A}\}$ can be defined as:

$$\Re\{\mathbf{A}\} = \{\mathbf{A} | -\Delta_\mathbf{A} \preceq \mathbf{A} - \hat{\mathbf{A}} \preceq \Delta_\mathbf{A}\} \qquad (6)$$

where $\preceq$ denotes element-wise inequality and for all Δk_{ij} from Eq. (3) the matrix $\Delta_\mathbf{A}$ is defined as

$$\Delta_\mathbf{A} = \begin{bmatrix} -(\Delta k_{12} + \Delta k_{13} + \Delta k_{10}) & \Delta k_{21} & \Delta k_{31} \\ \Delta k_{12} & -\Delta k_{12} & 0 \\ \Delta k_{13} & 0 & -\Delta k_{31} \end{bmatrix}.$$

Polyhedrons $\Re\{\mathbf{B}\}$ and $\Re\{\mathbf{C}\}$ can be similarly defined.

To model different types of patient populations, in the Patient model each of the predefined parameters can vary up to 10% from its initial value (e.g., $\Delta k_{12} = 0.1 k_{12}$).[2] In this case, to check the safety condition from (4) it is not possible to use the time response from (5) since matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are unknown. To deal with this type of uncertainty, when matrices belong to a predefined set of polyhedrons, the following theorem can be utilized:

Theorem 1: Consider an LTI system with uncertain parameters of the form:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \qquad (7)$$
$$y(t) = \mathbf{C}x(t).$$

where $\mathbf{A} \in \Re\{\mathbf{A}\}, \mathbf{B} \in \Re\{\mathbf{B}\}$ and $\mathbf{C} \in \Re\{\mathbf{C}\}$. If a constant input vector u_i is connected to the system's input at time t_0 and $x_0 = x(t_0)$, then for all $t \geq 0$:

$$t \geq \frac{1}{||\tilde{\mathbf{A}}||} \ln \left(\frac{|y(t_0 + t) - y(t_0)|}{||\tilde{\mathbf{C}}|| \cdot (||x_0|| + \frac{||\tilde{\mathbf{B}} u_i||}{||\mathbf{A}_{min}||})} + 1 \right) \qquad (8)$$

where:

$$\tilde{\mathbf{A}} = \underset{\mathbf{A} \in \Re\{\mathbf{A}\}}{\operatorname{argmax}} ||\mathbf{A}||, \tilde{\mathbf{B}} = \underset{\mathbf{B} \in \Re\{\mathbf{B}\}}{\operatorname{argmax}} ||\mathbf{B} u_i||, \tilde{\mathbf{C}} = \underset{\mathbf{C} \in \Re\{\mathbf{C}\}}{\operatorname{argmax}} ||\mathbf{C}||$$
$$(9)$$
$$\mathbf{A}_{min} = \underset{\mathbf{A} \in \Re\{\mathbf{A}\}}{\operatorname{argmin}} ||\mathbf{A}|| \qquad (10)$$

Proof: See appendix A ∎

If $x_0 = x(t_0)$ is known, the bound for t_{crit} can be computed as shown in (8). However, in cases where only values $y(t_0)$ and $y(t_0 + t)$ are known it is not possible to exactly determine x_0 since $y(t_0) = Cx_0$ and matrix C might not be invertible (as in our case). In this case for a lower bound $\tilde{t}_{crit}$ we consider a vector norm $||\tilde{x}_0||$ that is an upper bound on the maximal norm vector x can take. To obtain this value we simulated the patient's dynamics when the patient parameters are initialized as $\mathbf{A} = \tilde{\mathbf{A}}$ and $\mathbf{B} = \tilde{\mathbf{B}}$ (from (9)), since this configuration at each time instance maximizes value for $||\dot{x}||$ from (1). Note that the proposed bound on $||x_0||$ is very conservative as it provides an upper bound on $||x(t)||$ in the general case (i.e.,

---

[2]Although we can not guarantee that the uncertainty bound (i.e., 10%) can be used to model all types of uncertainties in the Patient model, we have opted to use it as it provides a way to model different types of patients. However, if Patient's dynamics (i.e., pharmacokinetics) is better understood for particular types of patient populations, a new uncertainty bound can be easily adopted.

for the reachable domain) and not only in points where $y(t_0) = Cx_0$. However, as we will see later, even with this bound it is possible to guarantee the safety requirement.

When the pump is continuously delivering the drug it is possible to compute a lower bound for t_{crit} using Theorem 1. In this case, the safety conditions will be satisfied if:

$$t_{POdel} + t_{net} + t_{Sup} + t_{net} + t_{Pump} + t_{P2PO} + t_{pi} \leq \tilde{t}_{crit}$$

where:

$$\tilde{t}_{crit} = \frac{1}{||\tilde{\mathbf{A}}||} \ln \left(\frac{\frac{|\Delta H^{\text{SpO}_2}|}{\text{SpO}_{2_{gain}}}}{||\tilde{\mathbf{C}}|| \cdot \left(||\tilde{x}_0|| + \frac{||\tilde{\mathbf{B}}u_i||}{||\mathbf{A}_{max}||} \right)} + 1 \right) \quad (11)$$

and $\text{SpO}_{2_{gain}} = 0.35 \cdot 5.8 = 2.03$ is the gain used in linear mapping between drug level and SpO_2 level (from Figure 5).

The value for $||\tilde{\mathbf{A}}_{min}||$ can be easily obtained in polynomial time as the solution of the appropriate convex optimization problems (i.e., using existing, very efficient methods for convex optimization). To solve this optimization problem on the predefined regions $\Re\{\mathbf{A}\}$ we used CVX, a package for specifying and solving convex programs [13] and obtained the value $||\tilde{\mathbf{A}}|| = 7.34 \cdot 10^{-3}$. On the other hand, it is not possible to utilize efficient methods for convex optimization to obtain $||\tilde{\mathbf{A}}||, ||\tilde{\mathbf{B}}u_i||$ and $||\tilde{\mathbf{C}}||$, as $\mathbf{A}_{max}$, $\mathbf{B}$, $\mathbf{C}$ are arguments of the maximizations over convex functions. However, to compute these values the following theorem can be used:

Theorem 2: Consider the optimization problem:

$$\max_{\mathbf{A} \in \Re\{\mathbf{A}\}} ||\mathbf{A}||$$

where $\Re\{\mathbf{A}\}$ is defined as in (6).[3] Then, there exists an optimal solution $\mathbf{A}_{max}$ which is a vertex of polyhedron $\Re\{\mathbf{A}\}$.

Proof: See appendix B. ∎

From the previous theorem to compute $||\tilde{\mathbf{A}}||$ it is necessary to calculate $||\mathbf{A}||$ in all 2^{n^2} vertices of the polyhedron. In our case, after computing the norm in $2^9 = 512$ points, we obtained $||\tilde{\mathbf{A}}|| = 8.97 \cdot 10^{-3}$. Similarly, after computing the norm in 2 points for both $||\mathbf{B}u_i||$ and $||\mathbf{C}||$ it was obtained $||\tilde{\mathbf{B}}u_i|| = 0.0917$, $||\tilde{\mathbf{C}}|| = 1.100$. Furthermore, after matrices $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}$ are determined using the previously described approach we obtained the value $||\tilde{x}_0|| = 27.08$. Therefore, in our model $\tilde{t}_{crit} = 24.83 \ s \approx 0.41$ min. Note that this bound is significantly lower than t_{crit} calculated for the 'fixed' patient model (which was 9.68 minutes). As mentioned before, this bound can be improved if $||\tilde{x}_0||$ is calculated only at points where $Cx_0 = y(t_0)$. However, even with this conservative bound, the value for t_{crit} satisfies the safety requirement if communication delay imposed by the network can be bounded.

Remark 2: Note that the bound provided in (11) can be improved using the approach presented in [14]. In addition, this approach can be also used in cases where a non-constant drug flow is delivered to the patient. However, the flexibility comes with a higher computational cost required by this procedure. Finally, if we are given a complex patient model in Simulink (e.g., a nonlinear system with uncertain parameters or a nondeterministic system) from which it is not possible to analytically determine t_{crit}, simulation of the Patient's behavior can be employed (as seen in Figure 6). As simulation

results depend on input signals and the model's initial state, it is essential that the worst-case scenario is known. Otherwise, methods similar to the one from [15] can be used. ∎

3) Safety conditions for the second Supervisor design: As described in Section II, for the second Supervisor design the pump is configured to stay activated only for a predefined fixed duration of time. In addition, Supervisor stops the pump only if the patient is in the *Alarming* region. In this case, the safety requirement is satisfied if at least one of the two conditions are met. The first condition is described in (4) and it is equivalent to the safety condition for the first Supervisor design. The second safety requirement is that the PCA activation period, (i.e., duration of the drug flow) t_{dur}, satisfies the requirement that it can not drive the Patient from the *Safe* region to the *Critical* region. This condition can be described as $t_{dur} \leq t_{crit}$. Therefore, the second safety requirement is satisfied if:

$$t_{dur} \leq \tilde{t}_{crit}, \quad (12)$$

with $\tilde{t}_{crit}$ defined as in (11).

4) Limitations of system verification in Simulink: Currently there does not exist a holistic approach for verification of nondeterministic hybrid systems, with uncertain parameters, modeled in Simulink. Most of the existing tools use procedures like Instrumentation Based Verification [16] where monitors are designed to check whether the safety conditions are violated. However, since this approach utilizes coverage-based testing, it can be used only in combination with tools for automatic test data generation. Several such commercial (e.g., Reactis, Design Verifier) and non-commercial (e.g. [15]) tools exist. These tools do not guarantee full coverage of the design state-space.

On the other hand, from the safety conditions described in (4), (12) it can be concluded that safety of the closed-loop system can be mapped into analysis of timing relations between the system's components. Therefore, we have opted to use UPPAAL [3] for the verification of the closed-loop system. However, when UPPAAL is used for the system's verification it is necessary to guarantee that the patient model in UPPAAL over-approximates the patient's dynamics with respect to the utilized control algorithm (details are provided in the next section). Finally, it is worth noting here that UPPAAL also allows development of more detailed system models since it inherently supports composition of asynchronous components.

IV. VERIFICATION AND VALIDATION OF COMPONENTS AND SYSTEM IN UPPAAL

This section describes the UPPAAL model developed for the case study verification. The structure of the model follows the architecture of the system. For each component in Figure 2, the model includes a separate automaton. The automata communicate using synchronization channels and shared variables. Figure 7 shows the network of automata and communication between them. Solid arrows represent communication channels and dashed arrows represent shared variables.

A. UPPAAL Component Models

1) PCA Automaton: The automaton, which models the pump, is shown in Figure 8. When the pump is operational, it is either in the state running, with the shared variable

---

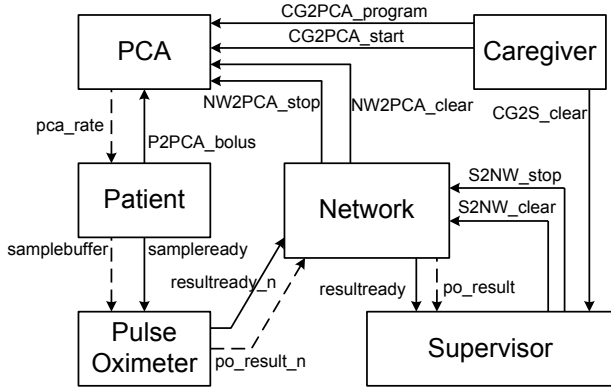[3]Note that matrix $\mathbf{A}$ does not have to be a square matrix.

Fig. 7. Communication structure of the UPPAAL model.

Fig. 9. Timed automaton for the network.

pca_rate set to default rate, or in the state bolusing, when pca_rate is increased by the bolus rate. Both rates are specified as parameters of the model. Furthermore, by setting the appropriate value for $time\_mode$ the pump can be set (at configuration, not runtime) to work in the both previously mentioned modes. In the first mode the pump is kept continuously on (i.e., in bolusing state) until stopped by the Supervisor, while in the second mode the pump can be bolusing for a fixed duration given by the value of the bolus_time parameter. The pump transitions to the bolusing state upon the signal received from the patient only if it is in the running state; in all other states, the signal is ignored. From either running or bolusing state, the pump can move to a stopped state (Rstopped or Bstopped, respectively) upon a signal from the network.

2) PO Automaton: The automaton, which represents the pulse oximeter, is shown in Figure 10. The operation of the automaton proceeds in rounds. Each round begins by setting the window size for the round based on the last sampled value. Then, the automaton collects the number of samples to fill the window. Samples are obtained periodically with the interval of 1 time unit, which corresponds to 100 ms. Finally, the result is stored in the po_result variable and delivered to the supervisor using the resultready channel.

3) Network and Caregiver Automata: The network is modeled using two automata. Communication from the Supervisor to the pump is modeled with automaton from Figure 9. The automaton implements a two-place buffer, which means that there may be two network messages in transit. The stop message may be dropped by the network, if the boolean parameter drop is set to true. We do not model dropping of the restart message, since the loss of these messages does not affect
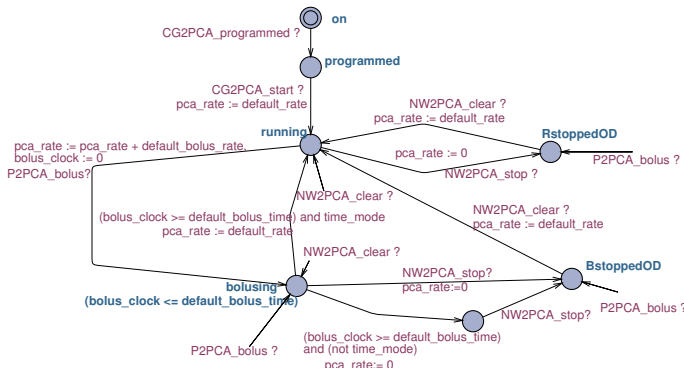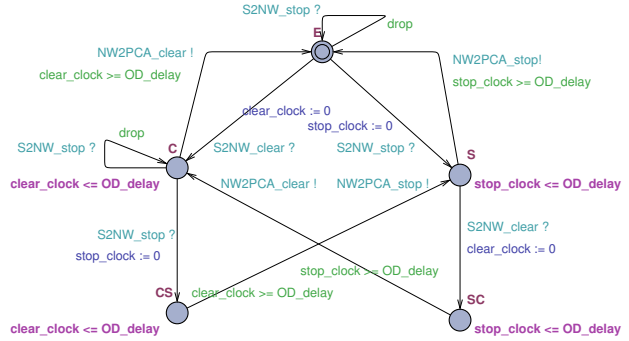
the safety of the patient. If messages are not dropped, they are delivered by the network in order. A similar automaton (not shown here) is used to introduce a bounded delay for PO measurements (i.e., po_result). Again, messages can be dropped if drop is set to true. The Caregiver automaton, not shown here, contains one state and can send any of the messages at any time. The messages include PCA pump and Supervisor configuration commands (e.g., CG2PCA_start, CG2S_clear). A more detailed model of the caregiver may include data-dependent behaviors, for example, the clear signal may be sent only if the SpO_2 reading is high enough. However, any other model will have fewer behaviors than the caregiver model used here, and thus the safety property will still hold.

4) Supervisor Automaton: The Supervisor automaton, shown in Figure 11, implements the simple control algorithm. Upon receiving a SpO_2 reading from the pulse oximeter, the supervisor compares it with the pre-defined threshold value and, if the result is too low, sends the stop message to the pump across the network. The model also incorporates a delay, which represents the worst-case execution time of the supervisor algorithm. Then, once the caregiver resolves the problem, the supervisor sends another message to restart the pump. For simplicity of the presentation, the Supervisor automaton only deals with SpO_2, not heart rates.

5) Patient Automaton: The Patient automaton, shown in Figure 12, periodically updates the drug level based on the flow rate of the pump and drug absorption rate. In addition, at any time, the model can deliver a sample as the function of the current drug level. The designed model implements a linear patient dynamics, which does not match the more realistic Simulink model. An important question that needs to be addressed is the consistency between the two models. Since the UPPAAL model of the controller uses significantly simplified patient dynamics, there is the possibility that verification results obtained on the UPPAAL model would not apply to the more detailed model and thus the system itself. Therefore, in the remaining of this subsection we address initialization of the Patient's and PCA pump automatons.
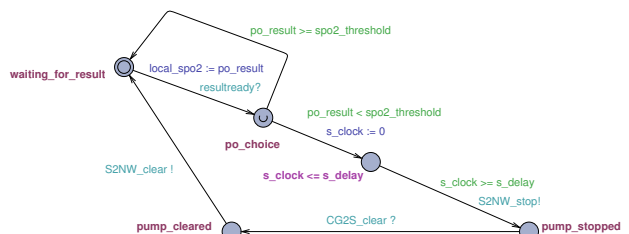

Fig. 8. Timed automaton for the PCA pump.

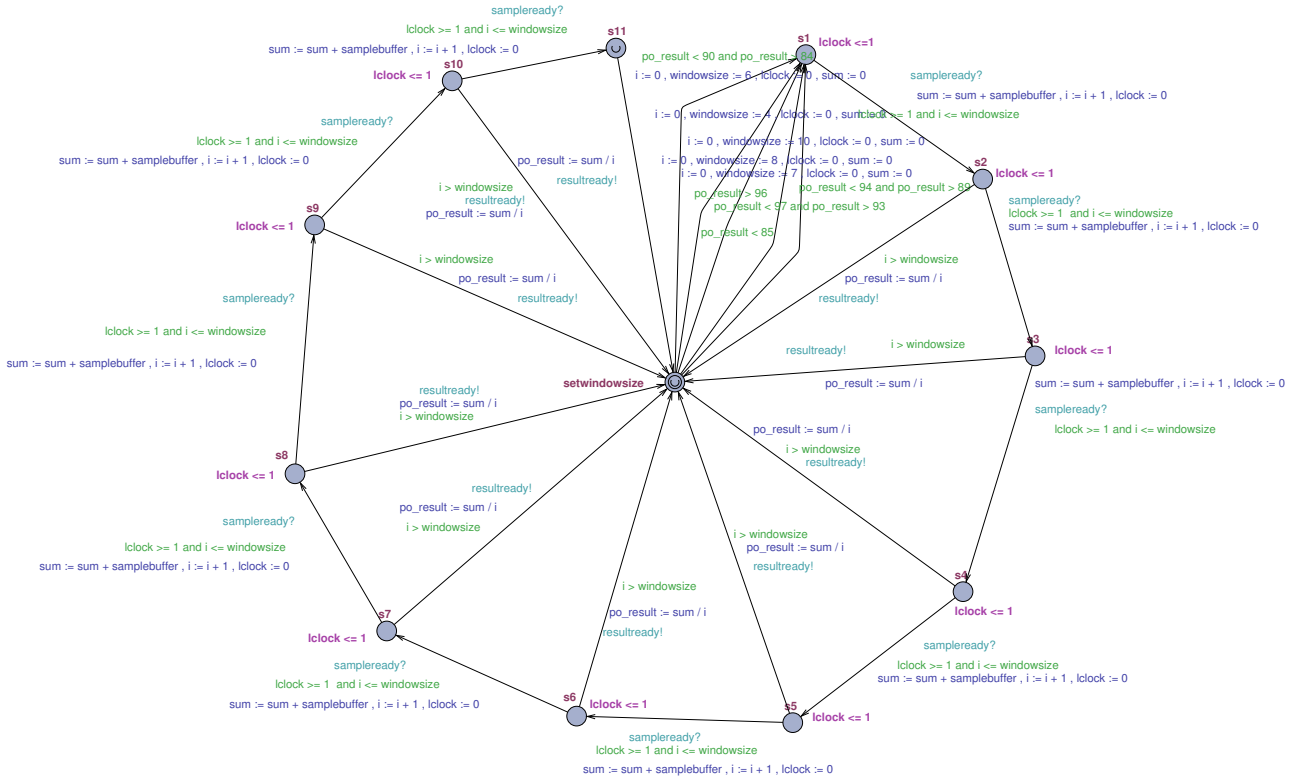Fig. 11. Timed automaton for the supervisor.

Fig. 10.   Timed automaton for the pulse oximeter.

6) Initializing model parameters: To determine values for the parameters of Patient and PCA pump automatons we consider the Patient's behavior with respect to the safety requirements. From the safety analysis provided in the previous section for both designed controllers (i.e., both Supervisor designs), it can be reasoned that the focus of interest for the safety conditions is with the patient's behavior in the *Alarming* region. In both of these cases it is required that the minimal time that the Patient can spend in the region is bounded by a value $\tilde{t}_{crit}$ (from (11)). Therefore, the value of $\tilde{t}_{crit}$, the minimal time it takes for the patient to be overdosed, is an important consistency check for the models.

In the UPPAAL model, the value of t_{crit} may be different (we denote it t_{crit}^U). This value can be obtained from the Patient model as:

$$t_{crit}^U = \left\lceil \frac{\Delta H^{\text{SpO}_2}/\text{SpO}_{2\,gain}}{PCA\_rateON - ab\_rate} \right\rceil, \qquad (13)$$

where $PCA\_rateON = default\_bolus\_rate + default\_rate$ and $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. To guarantee consistency between the UPPAAL and Simulink models, the value for t_{crit}^U has to be no greater than then the previously obtained value $\tilde{t}_{crit}$. Therefore, from condition $t_{crit}^U \leq \tilde{t}_{crit}$ and (13), we can derive the condition for the UPPAAL model
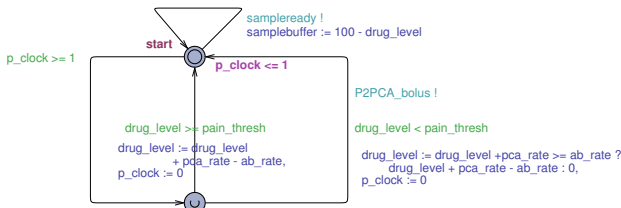

Fig. 12.   Timed automaton for the patient.

parameters ($default\_bolus\_rate$ and $default\_rate$ from PCA automaton and ab_rate from the Patient automaton).

In our case, t_{crit}^U turns out to be 20 seconds for the initial values of the parameters that described patient dynamics and drug flow ($pca\_rate = 3$, $ab\_rate = 2$), which is an order of magnitude smaller than $\tilde{t}_{crit}$. Clearly, the UPPAAL model overestimates the rate of change for the drug level in the patient body. And, since the system has been proven safe in this case, it would also be safe in a more realistic case. However, since we want to more realistically mimic the patient behavior in the *Alarming* region it is necessary to set the parameters in a way that:

$$\left\lceil \frac{\Delta H^{\text{SpO}_2}/\text{SpO}_{2\,gain}}{PCA\_rateON - ab\_rate} \right\rceil = \lfloor \tilde{t}_{crit} \rfloor \qquad (14)$$

This also allows relaxation of timing requirements imposed on other components in the control loop. For example, with this value it is possible to relax the value for the bound on network imposed delay (t_{net}) from Table I.

In addition, it is necessary to achieve consistency between UPPAAL and the physical patient model in cases when the pump is turned off. The goal is to minimize difference between the times (obtained from UPPAAL and Simulink models) that the patient would spend in the *Alarming* zone if the pump is turned off at the boundary of the *Critical* zone. Similarly as in the previous case, the condition that relates UPPAAL model parameters with the patient model from (1) can be obtained from (11) where $\mathbf{u}_i = 0$ (i.e., drug delivery is turned of):

$$\left\lceil \frac{\Delta H^{\text{SpO}_2}/\text{SpO}_{2\,gain}}{rate\_down} \right\rceil = \left\lfloor \frac{1}{||\tilde{\mathbf{A}}||} \ln \left(\frac{|\Delta H^{\text{SpO}_2}|/\text{SpO}_{2\,gain}}{||\tilde{\mathbf{C}}|| \cdot ||\tilde{x}_0||} + 1 \right) \right\rfloor \qquad (15)$$

where $rate\_down = ab\_rate - default\_bolus\_rate$. Finally, from (14), (15) the parameters for PCA and Patient automaton

in the UPPAAL model can be obtained.

B. Verifying PCA System Safety Properties

The main safety property that needs to be verified on the UPPAAL model is whether or not the patient can enter the *Critical* region, where SpO$_2$ and heart rate are low enough to indicate a respiratory arrest. Before verifying safety, however, we perform several auxiliary checks to ensure model sanity.

We express properties we verify in the subset of the Computational Tree Logic (CTL) [17] used by UPPAAL. The main temporal operators of this logic that we use are $A\Box\phi$, which means that ϕ is satisfied in every state along every execution path from the current state, and $A\Diamond\phi$, meaning that ϕ is satisfied eventually along every path.

The first sanity check is the absence of deadlocks in the model. Another sanity check is that once the SpO$_2$ level goes below the pain threshold, it eventually goes up. This property is captured by the temporal logic formula

$$A\Box(samplebuffer < pain\_thresh \Rightarrow$$
$$A\Diamond samplebuffer \geq pain\_thresh). \qquad (16)$$

Note that the property is defined in terms of the true SpO$_2$ level as defined by the patient model, not the sensor reading obtained by the supervisor. Intuitively, this property should hold, because the normal infusion rate is lower than the drug absorption rate. Once the patient stops requesting new boluses and the last bolus infusion is over, the drug level will start decreasing and thus SpO$_2$ and heart rate levels should increase, until they reach pain threshold again. Finally, we check that the pump is stopped if the patient ever enters the *alarming* region. Formally,

$$A\Box(samplebuffer < alarm\_thresh \Rightarrow$$
$$A\Diamond(PCA.Rstopped \lor PCA.Bstopped)). \quad (17)$$

We consider this property to be a sanity check rather than a safety requirement, because wrong parameters of the model – for example, too short bolus duration or too high drug absorption rate – can make the system appear safe (that is, SpO$_2$ level never goes too low), but it would be safe for the wrong reason. All sanity checks were passed by the UPPAAL model described above when no dropped messages are allowed. Clearly, property (17) does not hold if messages can be dropped.

Finally, we turn to checking the main safety property. With the threshold for the *Critical* region set to 70%, the property $A\Box(samplebuffer \geq critical)$ is satisfied if the stop message cannot be dropped. However, if losing messages is enabled in the network automaton, the property is not satisfied.

V. FAILURES AND FAIL-SAFE PCA SYSTEM

We have seen in Section IV that the system does not satisfy its safety property if network messages can be lost. Also, it can be observed from (4) that if any of the delays on the left side of the equation is significantly increased, the condition would not be satisfied. Increase in any of the delays can be caused by a component or network failures, which would result in an open-loop system.

To provide safety assurance realistic scenarios have to be taken into account, where network failures occur or PO accidentally gets detached from the Patient. Thus, the Supervisor's control algorithm and PCA pump's designs have to guarantee the system's *open-loop safety*. For the case study this denotes that even if the Supervisor does not receive the right values for HR and/or SpO$_2$ or the pump does not receive the command to disable the drug flow, the system design has to guarantee that the patient would not enter the *Critical* region. In addition, open-loop safety implies that even if the patient keeps pressing the button no drug flow will be enabled if there is a possibility that the amount of infused drug can harm the patient.

One way to design the system that complies with open-loop safety requirement is to make changes in the closed-loop system from Figure 2. In the modified system the pump receives activation command from the *Supervisor* (not the *Patient*) along with the duration of the drug flow (which does not have to be fixed). When the Patient presses the button if the Supervisor is informed about current HR and SpO$_2$ values it is able to determine a duration of the pump's activation (Δt_{safe}) that guarantees the patient's safety. An additional condition is that the Supervisor will disregard the button press during t_{del} units of time after the pump is stopped. Here, t_{del} takes into account all the delays in the loop and is defined as follows:

$$t_{del} = t_{POdel} + t_{net} + t_{Sup} + t_{net} + t_{Pump} + t_{P2PO} + t_{PCA2P}$$

The imposed condition ensures that the last drug delivery will take full effect before the next *'button pressed'* command is sent to the *Supervisor*. In addition, it implies that the drug level function will reach its local maximum before HR and SpO$_2$ measurements sent to the *Supervisor* are obtained. All constituents of t_{del}, except the network delay, can be calculated as described in the previous Section. We assume that if the message is delivered, the underlying real-time network provides a guaranteed bound on the network delay, as described in [5].

To calculate Δt_{safe} the following parameters are utilized:
- h_{cur} - last received SpO$_2$ level,
- Δh - worst case increase in SpO$_2$ level due to system latencies while the pump is off,
- $\Delta h(h_0, \Delta t)$ - maximal SpO$_2$ decrease cause by drug flow of the duration Δt when the initial SpO$_2$ level is h_0.

Therefore, to satisfy the safety requirement, Δt_{safe} has to meet the condition:

$$h_{cur} + \Delta h + \Delta h(h_{cur}, \Delta t_{safe}) \geq H_2^{\text{SpO}_2}. \qquad (18)$$

It is worth noting that although we use SpO$_2$ measurements to calculate Δt_{safe}, it is also possible to use HR measurements, as the boundaries of the *Critical* region (i.e., H_2^{HR} instead of $H_2^{\text{SpO}_2}$) correspond to same drug level values. The requirement, that after the pump is stopped, the Supervisor will disregard if the button is pressed for t_{del} time units, implies that when the button pressed command is accepted when $\Delta h \geq 0$ (as SpO$_2$ measurements decrease when drug level decreases), since the drug level will reach its local maximum before the measurements are taken. Therefore, the new safety condition can be obtained from:

$$\Delta h(h_{cur}, \Delta t_{safe}) \geq H_2^{\text{SpO}_2} - h_{cur} \qquad (19)$$

To determine Δt_{safe} Theorem 1 can be utilized. In this case Eq. (19) is satisfied if:

$$\Delta t_{safe} \leq \tilde{t}_{safe} = \frac{1}{||\tilde{\mathbf{A}}||} \ln \left(\frac{|H_2^{\text{SpO}_2} - h_{cur}|/\text{SpO}_{2\,gain}}{||\tilde{\mathbf{C}}|| \cdot \left(||\tilde{x}_0|| + \frac{||\tilde{\mathbf{B}}u_i||}{||\mathbf{A}_{min}||} \right)} + 1 \right) \quad (20)$$

Obviously if $h_{cur} > H_1^{\text{SpO}_2}$ ($H_1^{\text{SpO}_2}$ is the SpO$_2$ threshold value for the *Alarming* region) Δt_{safe} would be greater than $\tilde{t}_{crit}$. Similarly, if the patient is already in the *Alarming* region ($h_{cur} \leq H_1^{\text{SpO}_2}$) then $\Delta t_{safe} \leq \tilde{t}_{crit}$.

The system guarantees open-loop safety since, even if some (or all) of the messages are dropped, the patient would never enter the *Critical* region. The reason is that the pump activation command also contains the duration of the drug flow. In addition, if the flow durations are calculated properly, the pump will stop before safety requirement is breached.

To guarantee open-loop safety an assumption was made that the Supervisor is able to determine whether received measurements are valid. This is a reasonable assumption because modern POs send an invalid code in cases when valid measurements can not be obtained.[4] Note that in this work we do not consider failures where component's behavior differs from the behavior described by their models. For example, if the pump is active longer than requested, or if PO does not send an invalid code when it is unable to obtain valid measurements.

The presented solution is not the only way to guarantee open-loop safety. An alternative approach, which requires minimum change to existing PCA infusion pumps, is to have the Supervisor instruct the pump to pump the maximum amount of drugs that can be injected. Using a procedure similar to (20) the Supervisor can easily calculate the maximum allowed drug dosage. This system is also inherently fail-safe since the worst consequence of failed network, Supervisor, or PO is no drug injection, as in the previous case.

VI. RELATED WORK

Formal methods have traditionally been used for verification of time-critical and safety-critical embedded systems [18]. However, until recently they have not been used for medical device certification. Formal techniques have been applied to improve medical device protocols [19] and safety [20]. In addition, in [8] the authors present the use of Extended Finite State Machines for model checking of the Computer Automated Resuscitation A medical device. Nonetheless, although these papers consider relevant clinical application, in all of them either a simplified patient model was utilized or the patient was not modeled at all.

Continuous monitoring of the blood oxygenation of patients receiving PCA infusions has been done in the past and even commercially implemented. The Alaris 8210 SpO$_2$ Module connects to the Alaris 8000 pump controller and adds the ability to pause infusions based on a target SpO$_2$. Our approach shows how a similar system could be designed and validated. In particular, while the available commercial system is provided as a tightly-integrated system from a single vendor, our approach could be used to design and validate systems

based on devices from multiple sources as long as the timing and other necessary information is available.

VII. CONCLUSION

We have presented a clinically relevant case study of the closed-loop control of a PCA infusion pump. To perform safety analysis of the closed-loop medical system we have used a model-driven approach that combines simulation-based validation of a continuous-time system model in Simulink with formal verification of a more abstract model using timed automata and the UPPAAL tool. In this case, the key to keeping the two models consistent is to derive timing parameters of the system from the Simulink model and use these constants in the UPPAAL model.

For the case study, we have shown that the system is safe under no failure assumptions. We identify how to deal with some of the failures that manifest as unbounded delays. The proposed method is based on the well-known notion of timed lease used in fault-tolerant distributed systems. We believe that such a technique can be applied to other tightly integrated medical systems in which fail safe is essential.

The dynamics in the considered case study is relatively simple. This choice is made on purpose, to better present the steps in our approach, including the analysis of a system with uncertain parameters. Given the simple case study, we believe that our approach allows us to construct safety cases for regulatory approval of closed-loop medical systems. With more complicated dynamics, some of the steps become more difficult; in particular, the derivation of timing parameters for patient model with uncertain parameters may require more sophisticated methods and further research may be required.

In this paper, we highlight the need for model-driven safety analysis of the closed-loop medical systems and, on a specific case study, we emphasize potential problems that might occur. However, it it will be of interest to extend our approach to more general closed-loop medical scenarios.

APPENDIX A
PROOF OF THEOREM 1

Proof: Consider the LTI system from (7). The time response at time $t_0 + t$ for all $t \geq 0$ can be computed as ([21]):

$$x(t + t_0) = e^{\mathbf{A}t} x_0 + \int_{t_0}^{t_0+t} e^{A(t+t_0-\tau)} \mathbf{B} u(\tau) d\tau.$$

Therefore, if $u(t) = u_i$ for all $t \geq t_0$, where u_i is a constant vector, the system evolution can be described as:

$$x(t + t_0) - x_0 = (e^{\mathbf{A}t} - \mathbf{I}_n)x_0 + \mathbf{A}^{-1}(e^{\mathbf{A}t} - \mathbf{I}_n)\mathbf{B}u_i \quad (21)$$

where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is $n \times n$ identity matrix. Due to the fact that for matrices P, P_1 and Q of the appropriate dimensions $||PQ|| \leq ||P|| \cdot ||Q||$ and $||P + Q|| \leq ||P|| + ||Q||$, from (21) it is possible to provide the following bound:

$$|y(t + t_0) - y(t_0)| = |\mathbf{C}(x(t + t_0) - x_0)|$$
$$\leq ||\mathbf{C}|| ||x(t + t_0) - x_0|| \quad (22)$$

Now, consider the term $||e^{\mathbf{A}t} - \mathbf{I}_n||$. From the definition of the exponential function we can obtain:

---

[4]Detachment of the probes from the patient's body or movements of the patient's hand are main reasons why POs do not obtain valid values.

$$||e^{\mathbf{A}t} - \mathbf{I}_n|| = ||\sum_{k=0}^{\infty} \frac{(\mathbf{A}t)^k}{k!} - \mathbf{I}_n|| = ||\sum_{k=1}^{\infty} \frac{(\mathbf{A}t)^k}{k!}||$$
$$\leq \sum_{k=1}^{\infty} \frac{(||\mathbf{A}||t)^k}{k!} = \sum_{k=0}^{\infty} \frac{(||\mathbf{A}||t)^k}{k!} - 1 = e^{||\mathbf{A}||t} - 1$$

(23)

Similarly it can be shown:

$$||\mathbf{A}^{-1}(e^{\mathbf{A}t} - \mathbf{I}_n)|| \leq ||\mathbf{A}||^{-1}(e^{||\mathbf{A}||t} - 1) \quad (24)$$

From (22)-(24) it is possible to obtain a bound on the change in the system's output:

$$|y(t + t_0) - y(t_0)| \leq ||\mathbf{C}||(e^{||\mathbf{A}||t} - 1)(||x_0|| + ||\mathbf{A}||^{-1}||\mathbf{B}u_i||)$$

Therefore, from the previous equation:

$$t \geq \frac{1}{||\mathbf{A}||} \ln\left(\frac{|y(t_0 + t) - y(t_0)|}{||\mathbf{C}|| \cdot (||x_0|| + ||\mathbf{B}u_i||||\mathbf{A}||^{-1})} + 1\right) \quad (25)$$

for all matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ that belong to the polyhedrons $\Re\{\mathbf{A}\}$, $\Re\{\mathbf{B}\}$ and $\Re\{\mathbf{C}\}$ respectively.

Eq. (25) provides a lower bound on the value t for which the system output can progress from point $y(t_0)$ to the point $y(t_0 + t)$ for any values of the aforementioned matrices. The bound is a function of the patient parameters (i.e., matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$). The goal is to provide the bound value that is the minimum of the term on the right side of (25) for all matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ belonging to the corresponding polyhedrons.

The term on the right side of (25) decreases as the values $||\mathbf{A}||$, $||\mathbf{C}||$, $||x_0||$, $||\mathbf{A}||^{-1}$ and $||\mathbf{B}u_i||$ increase. Note that it is not possible to calculate the value for $||\mathbf{A}||$ that will minimize the term. Thus, for simplicity two independent problems are considered:

$$\max_{\mathbf{A} \in \Re\{\mathbf{A}\}} ||\mathbf{A}|| \quad \text{and} \quad \max_{\mathbf{A} \in \Re\{\mathbf{A}\}} ||\mathbf{A}||^{-1}$$

As $\max_{\mathbf{A} \in \Re\{\mathbf{A}\}} ||\mathbf{A}||^{-1} = \frac{1}{\min_{\mathbf{A} \in \Re\{\mathbf{A}\}} ||\mathbf{A}||}$, by defining $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}$ as in the Theorem statement, Eq. (8) can be obtained. ∎

APPENDIX B
PROOF OF THEOREM 2

Proof: Assume that there exist a matrix $\tilde{\mathbf{A}} = \text{argmax}_{\mathbf{A} \in \Re\{\mathbf{A}\}} ||\mathbf{A}||$, such that $\tilde{\mathbf{A}}$ is not a vertex and that there does not exist a matrix $\mathbf{A} \in \Re\{\mathbf{A}\}$ such that $||\mathbf{A}|| = ||\tilde{\mathbf{A}}||$. Since $\tilde{\mathbf{A}}$ is not a vertex and polyhedron $\Re\{\mathbf{A}\}$ is bounded there exists a matrix element $\tilde{a}_{ij}$ such that

$$\hat{a}_{ij} - \Delta a_{ij} < \tilde{a}_{ij} < \hat{a}_{ij} - \Delta a_{ij}. \quad (26)$$

Denote with $\hat{\mathbf{A}}_L$ and $\hat{\mathbf{A}}_H$ matrices whose all elements are equal to the appropriate elements from $\tilde{\mathbf{A}}$ except the element from i^{th} row and j^{th} column, and lets assume that these elements are $\hat{a}_{ij}^L = \hat{a}_{ij} - \epsilon$ and $\hat{a}_{ij}^H = \hat{a}_{ij} + \epsilon$, where $\epsilon > 0$ is chosen such that $\hat{a}_{ij}^L$, $\hat{a}_{ij}^H \in [\hat{a}_{ij} - \Delta a_{ij}, \hat{a}_{ij} + \Delta a_{ij}]$ (from (26) such ϵ exists). Since $\tilde{\mathbf{A}}$ is the unique minimization argument for the optimization problem then $||\tilde{\mathbf{A}}|| > ||\hat{\mathbf{A}}_L||$ and $||\tilde{\mathbf{A}}|| > ||\hat{\mathbf{A}}_H||$. Therefore:

$$||\tilde{\mathbf{A}}|| > \frac{1}{2}\left(||\hat{\mathbf{A}}_L|| + ||\hat{\mathbf{A}}_H||\right) \quad (27)$$

On the other hand since $||\mathbf{A}||$ is a convex function and from the fact that $\tilde{\mathbf{A}} = \frac{1}{2}(\hat{\mathbf{A}}_L + \hat{\mathbf{A}}_H)$ it can be concluded:

$$||\tilde{\mathbf{A}}|| \leq \frac{1}{2}\left(||\hat{\mathbf{A}}_L|| + ||\hat{\mathbf{A}}_H||\right)$$

which is in contradiction with (27). Thus, for the optimization problem there does not exist the unique minimization argument that is not a vertex, implying that there exist an optimal solution $\mathbf{A}_{max}$ that is a vertex of polyhedron $\Re\{\mathbf{A}\}$. ∎

REFERENCES

[1] *STAM 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.

[2] 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, pp. 194–199, 2005.

[3] G. Behrmann, A. David, and K. Larsen, "A tutorial on uppaal," in *Formal Methods for the Design of Real-Time Systems (revised lectures)*, ser. LNCS, vol. 3185, 2004, pp. 200–237.

[4] 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," *IEEE Computer*, vol. 39, no. 4, pp. 139–148, 2006.

[5] S. Fischmeister, J. M. Goldman, I. Lee, and R. Trausmuth, "Plug-and-play for medical devices: Experiences from a case study," *Biomedical Instrumentation & Technology*, vol. 43, no. 4, pp. 313–317, 2009.

[6] "Infusion pump improvement initiative," White Paper, Center for Devices and Radiological Health, U.S. Food and Drug Administration, April 2010.

[7] 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, no. Supplement 1, pp. 41–45, January 2008.

[8] D. Arney, R. Jetley, P. Jones, I. Lee, and O. Sokolsky, "Formal methods based development of a PCA infusion pump reference model: Generic Infusion Pump (GIP) project," in *HCMDSS-MDPNP'07: Proceedings of the of 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability*, 2007, pp. 23–33.

[9] V. Chan and S. Underwood, "A single-chip pulsoximeter design using the MSP430," Texas Instruments, Tech. Rep. SLAA274, Nov. 2005.

[10] B. W. Bequette, Ed., *Process control: modeling, design, and simulation*, 2nd ed. Prentice Hall, 2003.

[11] P. Jevon and B. Ewens, Eds., *Monitoring the Critically Ill Patient*, 2nd ed. Wiley-Blackwell, 2007.

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

[13] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming. http://stanford.edu/ boyd/cvx," June 2009.

[14] R. K. Yedavalli and C. R. Ashokkumar, "Time response bounds for linear parametric uncertain systems," *International Journal of Systems Science*, vol. 31, no. 2, pp. 177 – 188, 2000.

[15] R. Alur, A. Kanade, S. Ramesh, and K. C. Shashidhar, "Symbolic analysis for improving simulation coverage of simulink/stateflow models," in *EMSOFT '08: Proceedings of the 8th ACM international conference on Embedded software*, 2008, pp. 89–98.

[16] C. Ackermann, A. Ray, R. Cleaveland, J. Heit, C. Martin, and C. Shelton, "Model based design verification - a monitor based approach," in *Society of Automotive Engineers, World Congress*, 2008.

[17] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching time temporal logic," in *Workshop on Logic of Programs*, ser. LNCS, vol. 131, 1981, pp. 52–71.

[18] E. M. Clarke and J. M. Wing, "Formal Methods: State of the Art and Future Directions," *ACM Comput. Surv.*, vol. 28, no. 4, 1996.

[19] R. Alur, D. Arney, E. L. Gunter, I. Lee, J. Lee, W. Nam, F. Pearce, S. V. Albert, and J. Zhou, "Formal specifications and analysis of the computer-assisted resuscitation algorithm (CARA) Infusion Pump Control System," *Intl. J. Software Tools for Technology Transfer*, vol. 5, no. 4, pp. 308–319, 2004.

[20] A. ten Teije, M. Marcos, M. Balser, J. van Croonenborg, C. Duelli, F. van Harmelen, P. Lucas, S. Miksch, W. Reif, K. Rosenbrand, and A. Seyfang, "Improving Medical Protocols by Formal Methods," *Artif. Intell. Med.*, vol. 36, no. 3, pp. 193–209, 2006.

[21] W. J. Rugh, *Linear system theory (2nd ed.)*. Prentice-Hall, Inc., 1996.