Model-Based Dependability Analysis of Programmable Drug Infusion Pumps ⋆

Sriram Sankaranarayanan, Hadjar Homaei and Clayton Lewis.

University of Colorado, Boulder, CO. Email: first.lastname@colorado.edu

Abstract. Infusion pumps are commonly used in home/hospital care to inject drugs into a patient at programmable rates over time. However, in practice, a combination of faults including software errors, mechanical failures and human error can lead to catastrophic situations, causing death or serious harm to the patient. Dependability analysis techniques such as failure mode effect analysis (FMEA) can be used to predict the worst case outcomes of such faults and facilitate the development of remedies against them.
In this paper, we present the use of model-checking to automate the dependability analysis of programmable, real-time medical devices. Our approach uses timed and hybrid automata to model the real-time operation of the medical device and its interactions with the care giver and the patient. Common failure modes arising from device failures and human error are modeled in our framework. Specifically, we use "mistake models" derived from human factor studies to model the effects of mistakes committed by the operator. We present a case-study involving an infusion pump used to manage pain through the infusion of analgesic drugs. The dynamics of analgesic drugs are modeled by empirically validated pharmacokinetic models. Using model checking, our technique can systematically explore numerous combinations of failures and characterize the worse case effects of these failures.

1 Introduction

The delivery of critical medications through "smart" infusion pumps has become quite commonplace in home/hospital medical care. Yet, there have been numerous fatal or near-fatal incidents due to errors such as software error, pump malfunctions and human operator errors [1, 20, 24, 44, 46]. In this paper, we present a framework using formal verification techniques over timed and hybrid automata models to analyze the worst case effects of combinations of human and machine errors on the safety of the patient. Our framework systematically considers combinations of failure modes and provides quantitative predictions of the worst case outcomes for each combination. The failure modes considered here include mechanical failures in the pump, air bubbles, occlusions, empty vials, data entry errors and a failure to respond to alarms in a timely manner.

Our approach models the infusion pump by composing a simple hybrid automaton, incorporating a model of the pump, along the lines of the generic infusion pump model

⋆ This material is based upon work supported by the National Science Foundation (NSF) under award no. 1035845.

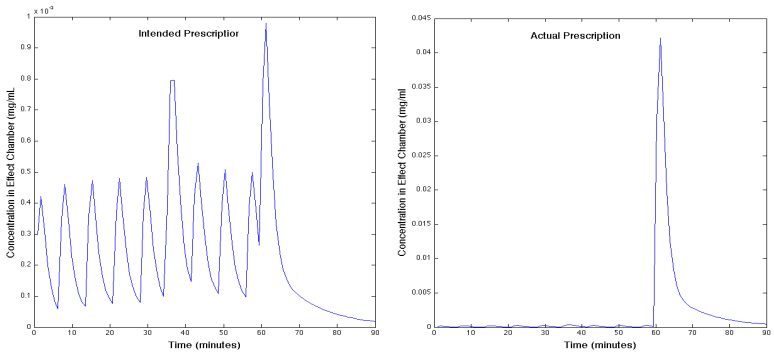


Fig. 1. Effect chamber concentration in the patient predicted by a pharmacokinetic model for (**left**) originally intended infusion (y-axis range: $[0, 1 \times 10^{-3}]$ mg/ml), and (**right**) actual infusion as a result of user error (y-axis range: $[0, 0.045]$ mg/ml).

proposed by Arney et al. [2] and the patient using a pharmacokinetic model that describes the concentration of the drug in various parts of the patient's body as the infusion proceeds over time [26, 39, 41]. Mechanical failures are modeled using failure mode variables and transitions. Our approach incorporates models of pump programming errors based on generic human error models commonly studied in the human factors community [22, 27, 36] along with the results of studies and reports on the common types of errors committed by caregivers while programming infusion pumps [3, 4, 28, 37, 47]. The composition of these models yields an affine hybrid automaton. The composed model is analyzed using bounded model checking (BMC) to compare the concentration of the drug in the original failure-free (original) model with the concentration in the failure-enabled model [5]. Specifically, the use of BMC allows us to search for executions, wherein, the timing of the faults and external disturbances yield the maximum (or minimum) possible concentration of the drug being infused in the patient's body. Comparing the range of drug concentrations resulting from the original fault-free model with the range obtained from the fault-enabled models provides a quantitative measure of the potential effects of the fault on the safety of the patient. Thus, the output of our analysis can be used as the starting point to help risk analysts construct dependability models in the form of a fault trees or failure mode effects analysis (FMEA) tables [17].

1.1 Motivation

We consider a brief summary of a fatal overdose incident due to wrong programming of an infusion pump, revealing some of the key hazards to patient safety due to programmable infusion pumps [20].

Event: The intended prescription ordered a 50 mcg/ml dose of the opioid (pain-killer) Fentanyl through a patient controlled analgesic (PCA) pump. A PCA pump allows the patient to request a preset dose of the drug by pressing a key. PCA pumps also enforce a set minimum lockout times between two requests. In this case, PCA request dose was 10 mcg with 6 minute lockout between doses. An extra bolus dose of 20 mcg (infused at maximum possible rate) was prescribed in case the pain level was high.

The device required as input: (a) the concentration of the drug stated in the vial, which was incorrectly entered as 1 mcg/ml instead of the correct concentration of 50 mcg/ml [1].This error may have resulted, in part, due to a "feature" in the device that silently reverted back to a previously entered value if an entry was not confirmed within some time limit; and (b) The demand dose for PCA was incorrectly set to 0.1 mcg instead of 10 mcg.

Outcome: Upon each PCA request the pump infused 0.1 mcg at 1 mcg/ml = 0.1 ml by volume. However, the actual drug concentration in the vial was 50 mcg/ml. This meant that the patient received 50 mcg/ml $\times$ .1 ml = 5 mcg upon each PCA demand dose. Fortunately, this was half the originally intended amount: the error in concentration had nullified that in the PCA demand dose. However, the pain persisted and a bolus dosage of 20 mcg was entered without correcting the error in the drug concentration. The pump infused 20 mcg at 1 mcg/ml = 20 ml of the drug into the patient. However, the patient actually received a dose of 20 ml $\times$ 50 mcg/ml = 1000 mcg of the drug, which resulted in a 50 $\times$ overdose. This overdose proved ultimately fatal.

Approach: Our approach can predict the scenario outlined as follows:

1. We consider the effect of numerous common pump programming errors on the data entered, as described in Section 4.
2. For each such error, our approach compares the intended vs. actual concentrations of the drug in the patient's body. To achieve this, we use hybrid automata models for the pump (Section 3.1) and pharmacokinetic models (Section 3.2).
3. Finally, we use bounded-model checking (BMC) using SMT solvers on a fixed time step approximation of the composed hybrid model to compare the possible range of drug concentrations in the original fault-free execution and the modified fault-enabled execution.

Figure 1 shows the concentration of the drug in the effect compartment of a pharmacokinetic model as a function of time both for the originally intended infusion and the infusion that occurs as a result of the error. The kinetics used for Fentanyl are as reported in the literature (Cf. Vuyk et al. [48]). Based on a comparison of the original with the faulty trace, it is seen that the new prescription results in a overdose of roughly 40 times the original concentration.

While the effects of an extreme scenario described above are easy to predict *qualitatively*, our framework provides three key advantages: It systematically explores a large range of possible operator mistakes and hardware/software failures. Secondly, our approach can explore the space of timings of the faults and the patient actions such as requesting a PCA bolus that can cause a worst-case outcome. Finally, our approach can quantitatively predict the worst-case outcome of a combination of faults on the patient using empirically validated pharmacokinetic models to predict the drug concentrations.

2 Preliminaries

We briefly recall the hybrid automaton model [21] and describe the use of formal verification to model faults and analyze dependability properties of dynamic systems [7].

[1] Note that recent models use barcode readers to obtain this information from the vial.

Hybrid automata are commonly used to model the composition of a discrete (software-based) controller interacting with a continuous environment [21].

Definition 1. *A hybrid automaton $\mathcal{H}$ is a tuple $\langle \boldsymbol{x}, \boldsymbol{d}, \mathcal{L}, \mathcal{T}, \mathcal{F}, \mathcal{I}, \Theta \rangle$, wherein,*

- $\boldsymbol{x} \in \mathbb{R}^n$ *refers to a vector of* continuous system variables.
- $\boldsymbol{d} \in \mathbb{R}^m$ *refers to a set of continuous external input (disturbance) variables.*
- $\mathcal{L}$ *refers to a finite set of* discrete locations *or modes.*
- $\mathcal{T}$ *refers to a set of discrete* transitions *(or mode changes). Each transition $\tau \in \mathcal{T}$ is of the form $\langle \ell, m, \rho_\tau[\boldsymbol{x}, \boldsymbol{d}, \boldsymbol{x}'] \rangle$, wherein, $\ell, m \in \mathcal{L}$ refer to the pre and post locations respectively, and ρ_τ is an assertion, representing the transition relation relation between the current state ($\boldsymbol{x}$) and the next state variables ($\boldsymbol{x}'$).*
- $\mathcal{D}$ *associates each location $\ell \in \mathcal{L}$ with a system of ordinary differential equations (ODEs) $\frac{d\boldsymbol{x}}{dt} = F_\ell(\boldsymbol{x}, \boldsymbol{d})$, wherein $F_\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a Lipschitz continuous function over $\boldsymbol{x}, \boldsymbol{d}$.*
- $\mathcal{I}$ *maps each location to a domain $\mathcal{I}(\ell) \subseteq \mathbb{R}^n$.*
- $\Theta \subseteq \mathcal{L} \times \mathbb{R}^n$ *refers to the initial conditions.*

The semantics and properties of hybrid automata are discussed elsewhere [21, 43].

2.1 Dependability Analysis

Dependability analysis of a safety critical systems identifies potential faults that may occur in the system and the worst-case effects of these faults on the safety, reliability and the performance of the system as a whole [17]. It has played a significant role in the design of safety critical control systems including avionics, satellites and nuclear reactors. Traditionally, dependability analysis techniques are *static* in nature, wherein the timing of the faults and the evolution of system state are not modeled.

Fault Tree: A fault tree is an enhanced circuit with logic gates (AND/OR/XOR) and other gates for modeling fault propagation, that computes the possibility of a particular type of system level failure as a function of the individual failure modes. Fault trees enable the computation of failure probabilities and expected time to failure as a function of the probability of the individual faults.

Failure Mode Effects Analysis: A *failure mode effects analysis* (FMEA) table lists different failure modes, mapping each failure mode to its causes, detection mechanisms, severity, expected frequency of occurrence and possible mitigations. FMEA is performed by risk analysts during system or process design.

In many complex systems, however, the timing of the events and the dynamics of the system are key to understanding the effects of failures. As a result, there has been much work on dynamic reliability techniques using a combination of continuous, discrete and stochastic models (including hybrid automata models [35]) in conjunction with techniques such as Monte-Carlo simulation for predicting failure probabilities [42].

Formal Dependability Analysis There has been much recent progress in the use of formal verification techniques such as model checking for automating dynamic dependability analyses [7, 8]. We provide a brief description of these approaches to model and reason about the effect of faults.

Fig. 2. (left) Examples of commercial infusion pumps and **(right)** major components of the drug infusion model.

Let $\mathcal{H}$ represent the model of the original system over state variables $\boldsymbol{x}$. The effect of failure is modeled by a combination of non-deterministic Boolean valued *failure mode variables* $f_1, \ldots, f_m$ and extra locations and transitions for modeling the system's operation upon failures to obtain a system $\mathcal{H}_f$ that can model the effects of failures. The primary purpose of the failure mode variables is to guard the transition to a subsystem modeling failure. Let $\varphi_1, \ldots, \varphi_k$ represent assertions that signify safety properties of the system $\mathcal{H}$. We assume that the original system satisfies these properties.

Symbolic model checking, or any reachability analysis technique, can be used on the augmented model $\mathcal{H}_f$ to search for reachable states that violate some of the safety properties [12]. Let $s \in [\![\varphi_j]\!]$ be a reachable error state in $\mathcal{H}_f$. Since $\mathcal{H}$ itself is assumed safe, we note that some subset F_s of the failure mode variables were enabled to reach the error state s. The overall analysis systematically collects such subsets F_s, which are termed cut-sets. Bozzano et al. also present techniques for finding minimal cut-sets of failure mode variables. The collection of failure modes that can lead a system to violate a property can be presented in the form of a fault tree or a FMEA table to help the panel of risk analysts better understand all the possible threats. This analysis has been integrated in the tool SLIM [8]. Our work is directly inspired by these efforts.

However, in our work, the modeling of human operator error is a key aspect of medical device failures. Mistakes such as data entry errors, unit conversion mistakes and misinterpretation of prescription labels modify the parameters of our model. Secondly, while it is possible to specify safety properties in terms of limits on how much drug can be infused, it is more natural in our setting to compare the range of doses that can be achieved using the original parameters (the intended infusion) vs. the range achieved by the fault-enabled parameters. Finally, we investigate pharmacokinetic models with dynamics that can be affine or non-linear.

3 System Models

Drug infusion pumps are commonly used in home/hospital medical care to inject drugs directly into the blood stream of a patient. Most infusion pumps are "programmable" by the caregiver, whose role consists of entering vital information from the prescription to the start of the infusion through a user interface. Infusion pumps support various *delivery modes*, as shown in Table 1. Combinations of these modes (eg., Bolus + PCA+

Mode	Description	Parameters
Continuous	Infusion at a continuous rate	rate cRate (mcg/min) concentration c (mcg/ml)
Bolus	Fast infusion of drug volume.	dosage w_b (mcg) concentration c (mcg/ml)
Patient-Control (PCA)	A bolus administered upon patient request	amtPerRequest (mcg) lockout (min) dose limit (mcg/hr) concentration (mcg/ml)

Table 1. Basic infusion modes supported by pump models along with the parameters relevant to each mode.

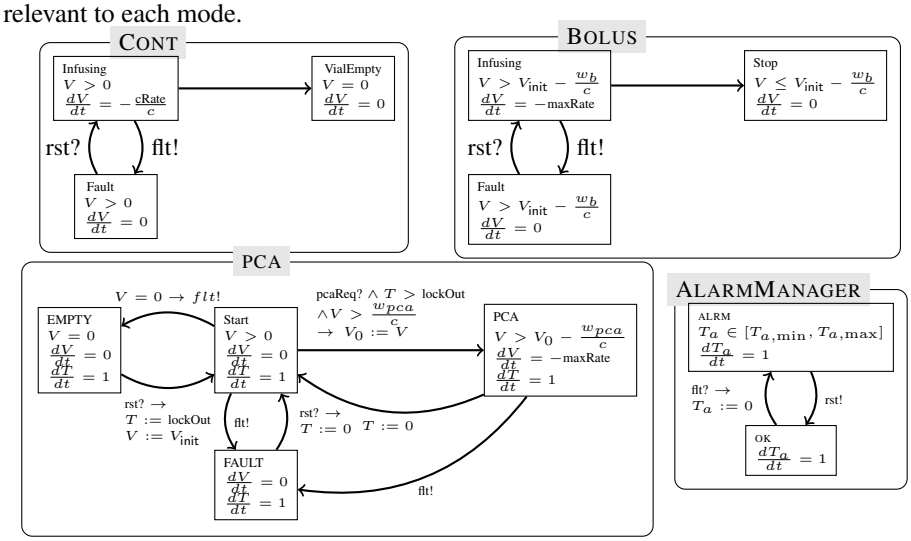
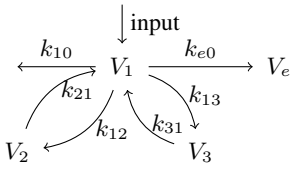


Fig. 3. Pump model for continuous, bolus, PCA infusion modes and the alarm manager component.

Cont. or Bolus + Cont.) are supported by many infusion pump models. The interface to the infusion pump includes protections from tampering such as a physical key to prevent reprogramming by the patient once the infusion has commenced.

The major components that are universally present in most pump models are: (1) A user-interface that allows the caregiver to "program" the infusion by entering the infusion mode and the data pertaining to each mode. (2) A syringe or bag loaded with a given total volume of the infused drug at some known concentration. (3) Various alarms that are sounded upon conditions such as airbubbles, blockages, pump failures, empty vials and so on. It is the responsibility of caregiver to reset the system to resume the infusion upon an alarm. (4) In Patient Controlled Analgesic (PCA) pumps, the patient can request a bolus of the drug by pressing a key.

Figure 2 shows some examples of infusion pumps and the three major components that we model as part of the dependency analysis of a drug infusion pump. We first discuss how various components are modeled in our approach and then present failure modes associated with the caregiver and the infusion pump itself. We refer the reader



Var	Remarks
V_i	Volume of the i^{th} chamber (ml).
$x_{1,2,3}$	Concentration in the i^{th} chamber (mcg/ml).
x_e	Concentration in the effect chamber (mcg/ml).
u	instantaneous infused concentration (mcg/min).
k_{ij}	kinetic diffusion param. from V_i and V_j.
k_{10}	param. for drug leaving chamber V_1.

Fig. 4. Three compartment pharmacokinetic model with an effect compartment. The kinetic parameters associated with each edge is shown.

to the work of Arney et al. as part of the generic infusion pump modeling project for a classification of existing models into many different types and their proposed generic (and comprehensive) infusion pump model [2]. In this work, we use a simplified version inspired by the Arney et al. infusion pump model. Our model is augmented with a patient model for the purposes of dependability analysis.

3.1 Pump Model

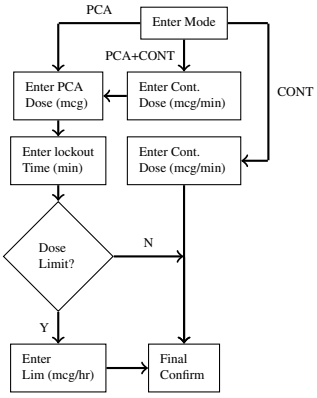
The pump itself is modeled by a hybrid automaton whose discrete modes describe the infusion mode. The continuous variables describe the rate at which the drug is leaving the vessel (and entering the patient) and the volume left in the vessel. We recall the three infusion modes as described in Table 1. The (simplified) hybrid automaton model for these modes are shown in Figure 3. We note that in each case, the pump component has a variable V representing the volume of the pump. Parameter V_{init} represents the volume in the syringe at the start of the infusion, w_b represents the ordered bolus weight and c represents the drug concentration, cRate the programmed rate of continuous infusion and maxRate the maximum possible rate at which the drug can be delivered (for a bolus/pca infusion). The infusion mode and the parameters w_b,c,w_{pca} and lockOut can be read in through barcode readers or entered through the user interface.

The pump issues internal fault events upon detection of a bubble, an occlusion or a pump failure. As a result, it transitions to a mode where the dynamics model the stoppage of the pump. These faults are transient in nature and result in an alarm being issued to the human operator. The operator then remedies the situation causing the fault and resets the operation back to the point in the infusion where the fault occurred. We assume in our model that all faults end up stopping the infusion. However, in many models, the infusion can often continue during some faults like low charge in the battery.

Alarm Manager: The alarm management model is shown in Fig. 3. We model a single alarm type that can be issued for various reasons. Our model assumes that alarms are handled within some fixed time interval by resetting the system to some fixed state. Our model uses timers to track the amount of time the current state is in a given mode. This allows us to model delays in the reset transition due to inattention to alarms.

3.2 Patient Model

The model of the patient is intended to capture the effect of the infusion in the patient's body. Since infusion pumps are able to deliver drugs at various pre-programmed con-



Prompt	Value
Enter Mode	PCA + Cont.
Enter Cont. Dose	0.1 mcg/min
Enter PCA Dose	1 mcg
Enter Lockout	5 min
Dose Limit?	Y
Enter Limit	10 mcg/hr

Fig. 5. (**Left**) Schematic interaction diagram for a medical infusion pump. Diagram is based roughly on the abbott PCA3 infusion pump. (**Right**) An example prescription entered through the interface.

centrations over time, we require a dynamical model of the drug as it is absorbed (and possibly metabolized) by the various cells in the patient's body and removed from the blood stream. We therefore use pharmacokinetic models to capture such effects over time [26, 39, 41].

A pharmacokinetic model uses multiple *compartments*, representing the various systems such as the vascular system, organs such as the liver, kidneys and the skin. The number of compartments used in a model depends on the specific drug whose kinetics are being modeled. The model is described by an ordinary differential equation whose variables represent the concentration of the drug in the various compartments. The dynamics of the model are given by

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_e} \end{pmatrix} = \begin{pmatrix} -(k_{10}+k_{12}+k_{13}) & k_{21}\frac{V_2}{V_1} & k_{31}\frac{V_3}{V_1} & 0 \\ k_{12}\frac{V_1}{V_2} & -k_{21} & 0 & 0 \\ k_{13}\frac{V_1}{V_3} & 0 & -k_{31} & 0 \\ k_{e0} & 0 & 0 & -k_{e0} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_e \end{pmatrix} + \begin{pmatrix} \frac{1}{V_1}u \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The values of the parameters may vary, depending on the drug whose dynamics is under investigation. Representative values are obtained as a result of studies conducted on a group of patients (Cf. [18, 30], for instance).

4 Human Interaction Models

In this section, we present a methodology for modeling human operator actions and the mistakes committed in the course of their interactions. In the setting of the infusion pump study, a human operator of the infusion pump is the care giver who is responsible for programming the infusion parameters at the start of the infusion and responding to alarms raised by faults such as air bubbles and occlusions in a timely manner.

Table 2. Major pump programming errors found in the literature.

	Error Description	References
1	Fields in the prescription are misinterpreted	[37, 40, 47]
2	Dosage Data Entry Error	[20, 37]
3	Unit Conversion Calculation Errors	[1, 4, 24, 28]
4	Pump mode selection errors	[3]

User Interface: Infusion pumps communicate with human operators by means of an user interface that displays messages to prompt the human operator and allows the user to enter values. Due to a lack of standardization of such interfaces, different devices offering the same set of basic functionalities may present a variety of interfaces for the same task [2]. Figure 5 shows an example interaction flowchart for an infusion pump based on an existing commercial model.

Prescriptions: Infusions of analgesics such as Fentanyl or morphine are tightly regulated, often requiring a prescription that carefully specifies the allowed dosage and the recommended mode of infusion. The details of a how a given infusion is prescribed can vary significantly, in general [4].

Example 1. An example prescription may call for a basal continuous dosage of 6 mcg/hr (=.1 mcg/min) of 50 mcg/ml of morphine and a PCA delivery of 1 mcg per request with a lockout interval of 5 minutes with an overall limit of 10 mcg each hour. It is the role of a care giver (often a registered nurse) to program this prescription using the infusion pump interface as shown in Figure 5. The data entered by the user is also shown.

4.1 Modeling Operator Mistakes

We discuss common mistakes that are committed by the operator during the course of an interaction and present a generic modeling system for these interaction errors. In theory, any *unintentional* deviation from a correct sequence of actions needed to deliver a given prescription of a drug to a patient can be classified as a mistake. There have been extensive studies by human factors experts and psychologists on the cognitive factors underlying human error [22, 23, 36]. Furthermore, the specific types of mistakes that operators commit while operating an infusion pump have also been well-studied and documented [3, 9, 20, 24, 28, 37, 40, 47]. Table 2 lists some of the common pump programming errors as reported in the literature. A comprehensive list of many types of hazards (including human error) for a generic infusion pump model has been put together by Arney et al. [3].

Data Entry Errors: In general, data entry errors depend on the type of interface used by the pump. For instance, empirical studies have shown that for pumps with keypads, the most harmful keypad entry errors include decimal point errors. Therein, a decimal point may either be inserted or deleted from the number entered due to the proximity of the decimal point key or variations in its placement across different interfaces [44]. Quirks in the interface such as reverting to a default value or a previously entered prescription also contribute to data entry errors (Cf. Section 1.1).

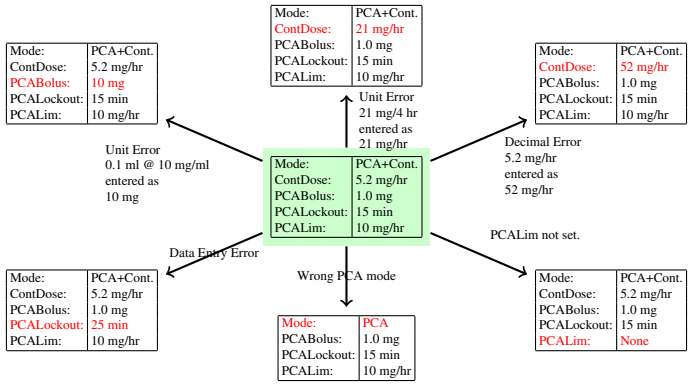


Fig. 6. Transformation of a prescription (shown center) through various mistakes.

Unit Conversion Errors: Prescriptions may be provided in units that differ from those required by the interface, requiring a calculation to be carried out for unit conversion. As a common example, an interface may require inputs in `mg/hr` while the prescription orders a dosage in `mcg / kg/min`. The conversion from the latter requires multiplying by the patient weight in `kg` and multiplying by 60 to convert from `min` to `hr` and finally dividing by 1000 to convert from `mcg` to `mg` [1, 4]. Some "calculator pumps" enable caregivers to carry out this calculation using the interface. Other types of pumps can accept inputs in one of many different choice of units. Yet, these models are not without hazards. A common error occurs when the difference between the originally prescribed units and the entered units is simply ignored, or the wrong units are ascribed to a dose. For example, a prescription that calls for `X mg` over `Y days` can be entered as `X mg/hr`, leading to a 24 fold overdose [24].

This leads to three types of errors: (a) the wrong calculation may be carried out (eg., divide by 60 instead of multiplying), (b) the right calculation may be carried out to yield a wrong result nevertheless or (c) the difference in units may be ignored and the original value entered as is. Unit conversion errors have been well documented especially in the context of infusion pump programming errors. Bates et al. presents a detailed study of the variability in the units used to prescribe dosages [4]. Furthermore, Lesar reports on a study of such unit conversion calculations to characterize the types of errors, their rates and effects [28].

Mode Selection Errors: It is often possible that pump can be misprogrammed by choosing the wrong delivery mode in the first place. A common example in PCA pumps is the choice of a "PCA only" mode where a "PCA+Cont." mode was ordered in the first place. Often, a wrong mode selection error can be detected during data entry if the pump prompts the user for data that is not part of the original prescription. The presence of too much redundant data in the prescription label is often a factor. This phenomenon is discussed in the context of infusion pumps by Thimbleby [46].

4.2 Mistake Models

We now describe a simple framework to model the effect of operator mistakes that will be used later in our overall analysis framework to predict worst case outcomes of human operator error. We may view the effects of a mistake simply as a *transformation* between the *intended* prescription and the *actual* prescription that results due to the mistake being committed [19, 22, 36]. Therefore, we simulate pump programming mistakes through set of transformations, wherein each transformation takes the original prescription as input and yields a set of possible "mistaken prescriptions" that could result, as if the mistake were actually committed. We illustrate this through an example.

Example 2. Figure 6 shows various transformations of an original prescription due to the errors such as mode selection error, data entry errors, calculation errors and other pump programming errors. For convenience, we have shown transformations due to single mistakes. However, it is possible to consider multiple mistakes as part of our mistake model by composing transformations.

5 Analysis Framework

We will now describe the framework used to explore possible worst case scenarios that may occur due to the presence of faults due to mechanical failures and human error.

The inputs to the analysis include the model of the pump, the patient and the data for the original prescription. The overall analysis has two parts: (a) Identify all failure modes. Machine/human errors in the model are identified by analyzing the process of prescribing the drug and the interface used to enter the data into the infusion pump; and (b) For each combination of k or less failure modes, we use model checking to search for runs of the model that exhibit the maximum values of the drug concentrations. Specifically, the model checker searches in the space of possible failure times for the various enabled failure modes and the timing of the PCA requests by the patient.

Analysis of Failure Mode Combinations: The overall idea behind our analysis framework is to compare the range of drug concentrations possible for the *reference model*, assuming the absence of faults and a *fault-enabled* model, wherein some pre-defined combination of faults are enabled. The focus of the comparison in our infusion pump case-study is the variable x_e in the patient model that represents the concentration of the drug in the effect compartment.

Fixed Time Step Approximation: Our analysis uses bounded model checking (BMC) [5] using SMT solvers [13, 31] on a fixed time step approximation of the model to find worst-case scenarios that can potentially maximize or minimize the value of x_4 at some time instant. We now briefly define the notion of a Δ time step approximation. We use some basic results for affine ODEs to discretize fixed time step approximations.

Definition 2 (Fixed Time step Runs). *Let $\mathcal{H}$ be an affine hybrid automaton. A run σ of $\mathcal{H}$ is said to conform to a fixed time step of Δ iff all discrete transitions in σ are taken at time instants that are integral multiples of Δ.*

Input:
Original Prescription Data.
Set of enabled failure modes.
Output:
Worst-case execution traces

1: Compute range of effect chamber concentrations over original prescription: $[x_{\min}, x_{\max}]$.
2: Simulate pump programming errors to create faulty prescription.
3: Create BMC encoding Ψ_M using faulty prescription parameters
4: Compute range $[y_{\min}, y_{\max}]$ for effect chamber concentration (using an SMT solver).
5: Compare $[x_{\min}, x_{\max}]$ with $[y_{\min}, y_{\max}]$ to compute worst-case deviations.

Fig. 7. Overview of Analysis Algorithm

A Δ time step approximation of a hybrid automaton for some time step $\Delta > 0$ considers only those runs σ that conform to a fixed time step of Δ.

It is easy to see that a fixed time step leads to an under approximation of the original event-triggered semantics. I.e, any (finite or infinite) run under the fixed time step approximation is also a run of the original system. On the other hand, there may be transitions that cannot be taken at integral multiples of Δ. Therefore, the choice of Δ needs to be small enough to retain most of the behaviors of the original system.

Assuming that the dynamics are of the form $\frac{d\boldsymbol{x}}{dt} = A\boldsymbol{x} + \boldsymbol{u}$, where A is invertible and the value of $\boldsymbol{u}$ remains constant during any time step interval, we conclude that

$$\boldsymbol{x}(t + \Delta) = M\boldsymbol{x}(t) + N\boldsymbol{u}, \text{ wherein } M = e^{\Delta A}, \text{ and } N = A^{-1}(e^{\Delta A} - I)\,.$$

Note that the pharmacokinetic models considered here satisfy the conditions required for the discretization.

Bounded Model Checking: Using a fixed time step Δ allows us to under approximate the original model in terms of a purely discrete model. Such a model is amenable to many verification techniques for discrete transition systems without the need to deal with the effects of ODEs, directly. For our study, we chose to use Bounded Model Checking (BMC) [5]. Using BMC, we encode the set of all runs of the model up to some fixed depth limit k by means of a logical formula in linear arithmetic. This formula is then solved by powerful SMT solvers such as Yices and Z3 [13, 16, 31].

6 Experiments

We report on an experimental evaluation of the ideas in this paper. The experimental evaluation focuses on the study of the infusion of the opioid Remifentanil through an infusion pump interface based roughly on existing commercial model — the Abbott PCA III infusion pump. Figure 8 shows the data fields entered through the pump interface and some of the calculations performed by the pump based on the entered data to derive model parameters.

The pump and the human patient models are as discussed in Section 3. We consider a set of human mistakes based on the interface used for data-entry as shown in Figure 9.

Data	Range	Value Used	Entry Method
DrugConc	50 mcg/ml	50	Barcode
Weight	[30-120] kg	60	Caregiver
PCAMode	{PCA-CONT, PCA-ONLY }	PCA-CONT	Caregiver
ContDose	[0 - 0.5] mcg/kg/min	0.1	Caregiver
PCABolus	[0.01 - 1] mcg/kg	0.5	Caregiver
Lockout	[5-20] min	10	Caregiver
DoseLimit	[3-10] req/hr or no lim	4	Caregiver

Field	Equation
CRate(ml/min)	$\begin{cases} \frac{\text{ContDose}*\text{Weight}}{\text{DrugConc}} & \text{if PCAMode} = \text{PCA-CONT} \\ 0 & \text{if PCAMode} = \text{PCA-ONLY} \end{cases}$
PCABolusVol(ml)	$\frac{\text{PCABolus}*\text{Weight}}{\text{DrugConc}}$
Lockout	As input
DoseLim	As input

Fig. 8. Data fields entered in a PCA infusion pump along with limits on the data values used in our simulations and (**bottom**) calculations performed to derive infusion pump parameters from entered data.

Mistake	Effect
Mode selection error (PCA-CONT vs. PCA-ONLY)	CRate' = 0
Weight entry error (weight in lb entered as kg)	CRate' = 2.2 * CRate PCABolusVol' = 2.2 * PCABolusVol
Weight entry error (weight kg entered as lb)	CRate' = CRate/2.2 PCABolusVol' = PCABolusVol/2.2
Unit Error (mcg/kg/hr entered as mcg/kg/min)	CRate' = 60 * CRate

Fig. 9. Possible data entry mistakes.

Implementation: We have implemented our analysis on top of a hierarchical, guarded command modeling language similar to existing tools such as SLIM that allows us to model the various components, faults and their interactions [8]. We compile and discretize our model and generate constraints to represent bounded depth executions. These constraints were solved using the solver Yices [16]. The maximization of the variable x_4 was performed by simulating a binary search over an interval. The minimum value achieved in our model is trivially 0 at the starting state. This procedure currently requires numerous calls to the solver. Our implementation along with the models used here are available upon request.

Model: The composed model of PCA pump with faults has roughly 6 modes, 11 parameters, 12 real-valued variables and 4 finite domain variables. Since machine faults are assumed to be transient, a counter `nFaults` is used in the model to store the number of machine faults to be considered in any execution. The initial value of this counter is varied from 0 (no fault) to 4 (four faults happen at non-deterministic time instances).

Table 3. Analysis results showing the influence of user error and machine faults on the predicted maximum drug effect chamber concentration. **nFaults:** Number of transient faults in a trace, **Conc:** maximum concentration achieved in ng/mL and **Time:** model checker time (seconds).

Mistake	nFaults = 0		nFaults = 1		nFaults=2		nFaults=4	
	Conc.	Time	Conc.	Time	Conc.	Time	Conc.	Time
None	6.1	64	5.4	447	5.1	1219	4.8	1218
Mode Selection Err.	3.1	16	3.1	213	2.9	500	2.8	459
Unit Selection Err.	150	78	149	210	132	182	101.2	1474
Weight (lbs as kgs)	13.4	73	12.6	1011	11.5	1306	10.5	2250
Weight (kgs as lbs)	2.7	41	2.6	534	2.4	1492	2.2	1617

Table 4. Analysis over 10 randomly selected prescriptions over the range described in Figure 8. The maximum concentration of reference model is normalized to 1. All figures rounded to one significant digit after decimal point.

Mistake	nFaults = 0			nFaults = 1			nFaults=2			nFaults=4		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
None	1	1	1	1	1	1.1	1	1	1.1	0.9	0.8	1.0
Mode Selection Err.	0.2	.1	4	0.3	0.1	0.4	0.3	.2	.4	.2	.1	.4
Unit Selection Err.	23	14	36	22	10	35	23.4	10	33	14.2	14.2	14.2
Weight (lbs as kgs)	2.0	1.8	2.3	2.2	2.1	2.5	2.2	2.1	2.4	2	1.8	2.3
Weight (kgs as lbs)	.4	.3	.4	.4	.4	.4	.4	.4	.5	.4	.4	.4

Table 3 shows the results of the analysis of various fault combinations in terms of the maximum concentration of drug achieved and the time taken by the model checker to complete the analysis. Comparing the maximum drug concentration of $6.1ng/mL$ predicted for the reference model (no human/machine faults) against the maximum values for other fault combinations yields insights into the effects of various faults. For instance, it is clear that mode selection errors roughly halve the maximum concentration achievable. The presence or absence of machine faults has little or no impact on the maximum. On the other hand, unit errors cause a 23 fold increase in the maximum effect chamber concentration achievable.

Overall, the results predicted by model checking agree well with our qualitative predictions of the outcome of the faults. However, the results are more useful since they can provide quantitative predictions and allow us to compare/rank faults based on the severity of their effects. For instance, the under dosage caused by machine faults is predicted to be less significant than that caused by a mode selection error or a unit error wherein the weight in kgs is treated as if it were in lbs. Note that under doses as well as over doses can have serious consequences, since restoring control of pain after an under dose can be difficult.

Finally, we investigate how the overall trends vary with the prescription data. Table 4 summarizes the results of repeating the experiment with 10 different randomly chosen prescriptions. The range of values for various parameters is described in Figure 8. In each case, the maximum value found for the reference execution is normalized to 1, thus allowing us to compare the results from different prescriptions. Table 4 shows the result of this comparison on 10 different randomly generated prescriptions. We note that

the overall direction of the effect of error seems independent of the actual prescription data themselves. Furthermore, in many cases, the magnitudes are quite similar.

7 Related Work & Conclusions

A vast body of work has focused on modeling and verification of functional correctness properties for user interfaces (Cf. [14, 33, 34], for instance). Much of this work has focused on the search for specific topologies in the state diagram of the interface that can indicate the presence of serious faults such as *mode confusion* [25, 29, 38]. The problem of characterizing various forms of human error has been studied in detail by cognitive scientists and human factors experts [15, 36]. In this regard, the use of generic mistake models to transform a given ideal interaction into a faulty interactions has been proposed in the past, notably by Hollnagel [22]. Fields presents an integration of this approach with formal verification tools to model and reason about operator error [19].

Formal and Semi-formal techniques have been applied to analyze infusion pumps for the presence of hidden modes and inconsistencies in the interface transitions [44, 45]. Bolton and Bass present a detailed model of the user interaction with an infusion pump by extending the operator function model framework. Their model lends itself to analysis using formal verification tools [6]. Bolton et al. apply their approach to provide a detailed model of the interface and various user actions required to program an infusion. However, in this paper, we consider a detailed model of the pump's operation and the dynamics of the drug concentration in the patient while summarizing the effect of user's mistakes using a mistake model. A combination of the two approaches seems quite desirable. However, it is quite likely that the analysis of such a detailed combined model may be intractable.

Model-based techniques remain popular in many domains including the development of reliable and safe user interfaces [10, 11, 32]. A common approach to the model-based testing of UI applications requires the construction of *mental models* of the user to characterize ways in which the interface can be used [10, 11]. However, these approaches seek to test the functional correctness of the interface itself, while operator errors are usually ignored.

In conclusion, we present a framework for dependability analysis of infusion pumps. In the future, we wish to extend this framework to study other types of programming errors in devices such as implantable pace-makers and total intravenous anesthesia. The model results also support exploration of user interface modifications that could guard against likely errors. One approach could be to incorporate logic into the pump controller to recognize possible user mistakes that may result in underdoses or overdoses. Since the appropriate dose depends on patient weight, so that the doses appropriate for adults are quite different from those for children, for example, it is not possible to specify definitely what the allowable range of doses is, within wide limits. Another suggestion could be to build closed loops to prevent mishaps by estimating drug concentrations using pharmacokinetic models and observable signals such as the respiration rate and blood oxygen content.

Bibliography

[1] Anonymous (Alberta, R.N.). Lack of standard dosing methods contributes to i.v. infusion errors. *Institute for Safe Medication Practices (ISMP) Medication Alert*, 64(4), April 2008.

[2] 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 *Proc. High Confidence Medical Devices, Software Systems and Medical Device Plug and Play Interoperability*, 2007.

[3] D. E. Arney, R. Jetley, P. Jones, I. Lee, A. Ray, O. Sokolsky, and Y. Zhang. Generic infusion pump hazard analysis and safety requirements: Version 1.0, 2009. CIS Technical Report, University of Pennsylvania. Available on-line: `http://repository.upenn.edu/cis_reports/893`, Accessed May 2011.

[4] D. W. Bates, T. Vandervreen, D. Seger, C. Yamaga, and J. Rothschild. Variability in intravenous medical practices: Implications for medication safety. *J. Joint Commission on Accredication of Healthcare Organizations*, 31(4):203–210, April 2005.

[5] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS*, volume 1579 of *LNCS*, pages 193–207, 1999.

[6] M. L. Bolton and E. J. Bass. Formally verifying human-automation interaction as part of a system model: limitations and tradeoffs. *Innovations Syst. Softw. Eng.*, 6: 219–231, 2010.

[7] M. Bozzano, A. Cimatti, and F. Tapparo. Symbolic fault tree analysis for reactive systems. In *ATVA*, volume 4762 of *LNCS*, pages 162–176. Springer, 2007.

[8] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, and M. Roveri. The COMPASS approach: Correctness, modelling and performability of aerospace systems. In *SAFECOMP 2009*, volume 5775 of *Lecture Notes in Comp. Sci.*, pages 173–186. Springer, 2009.

[9] J. L. Brady. First, do no harm: Making infusion pumps safer. *Biomedical Instrumentation & Technology*, 44(5):372–380, September 2010.

[10] P. A. Brooks and A. M. Memon. Automated GUI testing guided by usage profiles. In *Prof. ASE'07*, pages 333–342. IEEE Press, 2007.

[11] V. Chinnapongse, I. Lee, O. Sokolsky, S. Wang, and P. Jones. Model-based testing of GUI-driven applications. In *Prof. SEUS'09*, volume 5860 of *LNCS*, pages 203–214. Springer, 2009.

[12] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[13] L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[14] A. Degani and M. Heymann. Formal Verification of Human-Automation Interaction. *Human Factors*, 44(1):28–43, 2002.

[15] S. Dekker. *The Field Guide to Understanding Human-Error*. Ashgate Publishing, 2006.

[16] B. Dutertre and L. de Moura. The YICES SMT solver. Cf. `http://yices.csl.sri.com/tool-paper.pdf`, last viewed Jan. 2009.
[17] C. E. Ebeling. *Introduction to Reliability and Maintainability Engineering*. Waveland Inc., 2005.
[18] T. Egan, H. Lemmens, P. Fiset, D. Hermann, K. Muir, D. Stanski, and S. Shafer. The pharamcokinetics of the new short acting opioid remifentanil (G187084B) in healthy adult male volunteers. *Anesthesiology*, 74:881–892, 1996.
[19] R. Fields. *Analysis of erroneous actions in the design of critical systems*. PhD thesis, University of York, January 2001.
[20] M. Grissinger. Misprogram a PCA pump? it's easy!, July 2004. ISMP Medication Safety Alert., Accessed May 2011.
[21] T. A. Henzinger. The theory of hybrid automata. In *LICS'96*, pages 278–292. IEEE, 1996.
[22] E. Hollnagel. *Human Reliability Analysis Context and Control*. Computer And People Series. Academic Press Inc., San Diego, CA, 1993.
[23] E. Hollnagel. *Cognitive Reliability and Error Analysis Method*. Elsevier, Institutt for Energiteknikk, Halden, Norway, 1998.
[24] Institute for Safe Medication Practices Canada. Fluorocil incident root-cause analysis, 2007. Available on-line from `http://www.cancerboard.ab.ca/NR/....`
[25] A. Joshi, S. P. Miller, and M. P. Heimdahl. Mode confusion analysis of a flight guidance system using formal methods. In *22nd IEEE Digital Avionics Systems Conference (DASC'2003)*, October 2003.
[26] A. Kallen. *Computational Pharmacokinetics*. Chapman & Hall, 2007.
[27] B. Kirwan. *A Guide to Practical Human Reliability Assessment*. Taylor & Francis, 1994.
[28] T. S. Lesar. Errors in the useof medication dosage equations. *Archives of Pediatric Adoloscent Medicine*, 152:340–344, 1998.
[29] N. G. Leveson and E. Palmer. Designing automation to reduce operator errors. In *IEEE Trans. on Systems, Man, and Cybernetics*, page 7, October 1997.
[30] D. A. McClain and C. C. Hug. Intravenous fentanyl kinetics. *Clinical Pharmacology & Therapeutics*, 28(1):106–114, july 1980.
[31] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to DPLL(*T*). *J. ACM*, 53(6):937–977, 2006.
[32] A. Paiva, J. C. P. Faria, N. Tillmann, and R. F. A. M. Vidal. A model-to-implementation mapping tool for automated model-based GUI testing. In *ICFEM*, volume 3785 of *LNCS*, pages 450–464. Springer, 2005.
[33] P. Palanque. *Formal Methods in Human-Computer Interaction*. Springer-Verlag New York, Inc., 1997. ISBN 3540761586.
[34] F. Paternó and C. Santoro. Integrating model checking and HCI tools to help designers verify user interface properties. In *DSV-IS'00*, volume 1946 of *LNCS*, pages 135–150. Springer, 2001.
[35] G. Pérez-Castañeda, J.-F. Aubry, and N. Brinzei. Stochastic hybrid automata model for dynamic reliability assessment. *Journal of Risk and Reliability*, 225 (1):28–41, 2011.

[36] J. T. Reason. *Human Error*. Cambridge, UK: Cambridge University Press, 1990.

[37] J. Rothschild, C. Keohane, E. Cook, E. Orav, E. Burdick, S. Thompson, J. Hayes, and D. Bates. A controlled trial of smart infusion pumps to improve medication safety in critically ill patients. *Critical care medicine*, 33(3), 2005.

[38] J. Rushby. Using model checking to help discover mode confusions and other automation surprises. In *Proc. HESSD'99*, June 1999.

[39] V. Sartori, P. M. Schumacher, T. Bouillon, M. Luginbuehl, and M. Morari. Online estimation of propofol pharamacodynamic parameters. In *Proc. Conference on Engineering in Medicine and Biology*, pages 74–77. IEEE Press, 2005.

[40] J. Schein, R. Hicks, W. Nelson, V. Sikirica, and D. Doyle. Errors in the postoperative period: Causes and prevention. *Drug Safety*, 32(7):549–559, July 2009.

[41] S. L. Shafer, L. C. Siegel, J. E. Cooke, and J. C. Scott. Testing computer-controlled infusion pumps by simulation. *Anesthesiology*, 68:261–266, 1988.

[42] N. Siu. Risk assessment for dynamic systems: An overview. *Reliability Engineering & System Safety*, 43(1):43 – 73, 1994.

[43] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[44] H. Thimbleby. Ignorance of interaction programming is killing people. *ACM Interactions*, pages 52–57, 2008.

[45] H. Thimbleby. Contributing to safety and due diligence in safety-critical interactive systems development. In *ACM SIGCHI, EICS'09*, pages 221–230, 2009.

[46] H. Thimbleby. Is it a dangerous prescription? *BCS Interfaces*, 84:5–10, 2010.

[47] P. L. Trbovich, S. Pinkney, J. A. Cafazzo, and A. Easty. The impact of traditional and smart pump infusion technology on nurse medication administration performance in a simulated inpatient unit. *Qual. Saf. Health Care*, 19:430–434, 2010.

[48] J. Vuyk, M. J. Mertens, E. Olofsen, A. G. Burm, and J. G. Bovill. Propofol anesthesia and rational opioid selection. *Anesthesiology*, 87(6):1549–2562, 1997.