Towards A Test and Validation Framework for Closed-Loop Physiology Management Systems for Critical and Perioperative Care

Farooq Gessa
Bucknell University
Lewisburg, PA, USA
fmg005@bucknell.edu

Philip Asare
Bucknell University
Lewisburg, PA, USA
philip.asare@bucknell.edu

Aaron Bray
Kitware Inc.
Carborro, PA, USA
aaron.bray@kitware.com

Rachel Clipp
Kitware Inc.
Carborro, PA, USA
rachel.clipp@kitware.com

S. Mark Poler
Geisinger Health System
Danville, PA, USA
mpoler@geisinger.edu

ABSTRACT

Many of medical devices come equipped with a communication interface. Over the years, there has been interest in leveraging these interfaces to add computers to the loop to aid in decision making and automatic application of interventions. Such systems, which we call Closed-Loop Assistants (CLAs), are intended to help clinicians manage the cognitive load that can arise as the complexity of patient management increases. We present an open-source framework for examining CLA-patient interactions through software simulations of the CLA with *in-silico* patients to enable early testing and validation of proposed physiology management strategies. We show how this framework can be used to test different strategies across a small patient population. Considering a patient population is important because inter-patient variability is one of the critical factors that can hamper the ability of a medical cyber-physical system like the CLA to meet its goals. The ability to explore this variability early in the design process therefore helps us in increasing robustness of the system.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Applied computing** → *Life and medical sciences*; • **Information systems** → Process control systems.

KEYWORDS

In-silico patients, closed-loop physiology management, testing and validation

1 INTRODUCTION

Management of physiology in critical (intensive care unit) and perioperative (surgery-related) settings requires the closed-loop process of monitoring patient state, deciding on appropriate interventions (including inaction), and applying the appropriate intervention. In many cases, multiple patient variables must be tracked and many different actions must be taken. Both patient monitoring and application of interventions usually require the use of various medical devices. Many of these devices come equipped with a communication interface. Over the years there has been interest in leveraging these interfaces to add computers to the loop to aid in decision making and automatic application of interventions. Such systems are intended to help clinicians manage the cognitive load that can arise as the complexity of patient management increases. We call these systems Closed-Loop Assistants (CLAs) since they aid the clinicians in their performance of the various closed-loop tasks (as opposed to replacing the clinician entirely, which we believe to be unrealistic). Figure 1 shows a conceptual picture of a CLA.

One compelling application area for CLAs is in telehealth, particularly in isolation or contaminated environments (e.g. when treating infectious disease like Methicillin-resistant Staphylococcus aureus (MRSA) and Ebola). This includes manual remote management of monitor alarms, reporting intervals, infusion rates or ventilator settings (since CLA technology also enables remote control of devices). Additionally, the benefits of closed-loop control of infusions could minimize the need to enter a hazardous location, reducing risk to the clinician and risk that containment will not be maintained. The remote control could be local, just outside the containment environment, or by an external critical care consultant at a remote site. In general care settings, this sort of technology can also be used to convert telehealth from current surveillance/remote patient monitoring, where even change of monitoring settings must be done at the bedside, into remote patient care, where a CLA under the supervision of a remote clinician can intervene rather than send

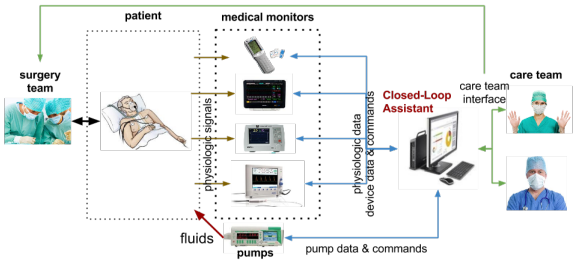


Figure 1: A conceptual picture of a CLA working with clinicians in an operating room.

Copyright held by the owner/author(s).
MCPS'18, April 2018, Porto, Portugal.

an alert to an on-site clinician who will require time to get to the bedside and orient to the patient's status. This can have a positive impact on patient safety.

Since the CLA is a medical cyber-physical system (MCPS), proper verification and validation is critical to ensuring safe and robust operation in the intended environment. The key behaviors to consider here are: (1) the CLA-patient interactions; (2) the CLA-clinician interaction; and (3) the CLA-plus-clinician-patient interaction to determine overall system efficacy. Being able to consider these interactions early in the design process allows various ideas to be tested in low-risk (and cost-effective) fashion to provide insights on how to improve designs.

Evaluation of systems with *in-silico* patients has been demonstrated to be useful for design of other MCPS [14, 23]. Encouraged by these developments, we are developing a framework for examining CLA-patient interactions through software simulations, where a CLA interacts with *in-silico* patients, to enable early testing and validation of proposed physiology management strategies. In this paper, we present the concepts behind the framework and describe its current incarnation. We show how this framework can be used to test different strategies across a small patient population. Considering a patient population is important because inter-patient variability can hamper the ability of MCPS to meet its goals. The ability to explore this variability early in the design process therefore helps us in increasing robustness of the system. The CLA case represents a more complex version of the problem than the previously-cited efforts because of the larger scope of applicability and resulting increase in variability in components.

This work is part of our larger goal of enabling system-in-the-loop evaluations ("system" because the CLA is a hardware-software system), where the CLA is instantiated as the intended hardware-software system, interacting with real medical devices, and with *in-silico* patients also capable of interacting with the real medical devices. Most importantly, this work is an open-source project. We believe that approaching the work in this manner would provide the greatest impact and benefit to the MCPS and medical community. Our repository can be found here: https://gitlab.bucknell.edu/fmg005/clasim.

2 THE FRAMEWORK

The framework, depicted conceptually in Figure 2, consists of two major pieces: a physiology platform that represents the patient (in our case Pulse [13]), and the CLA components consisting of models of medical devices (patient monitors and infusion pumps) and models of the physiology management algorithms. It currently models interactions as data/action flow between various components with the ability to inject real-world issues like time delays and data loss and inaccuracies in abstract fashion. Although Figure 2 shows a single physiology management algorithm, the framework can support multiple such algorithms simultaneously interacting with devices (and multiple independent CLAs interacting with the same patient).

The modular design allows for the exploration of variation that we are interested in. We can explore variation in any of the key components. For example, we can explore a specific instance of the CLA (devices plus algorithm) across many different *in-silico* patients. Or, for a specific patient, we can explore the effect of different versions of the CLA (by varying one or more of its components). Taken together, we can explore the impact of various CLA designs on clinical outcomes (which is what the CLA is supposed to help with) across a representative patient population, as represented in Figure 3), effectively running an *in-silico* trial in the early stages of development to identify promising design choices. This ability to explore the clinical impact of design choices has been advocated for in previous work on evaluating the efficacy [2] and safety [3] of systems, as well as the vision for MCPS [11] (called "High-Confidence Medical Devices Software and Systems" in the article). Below, we provide information on the still evolving parts of the framework.

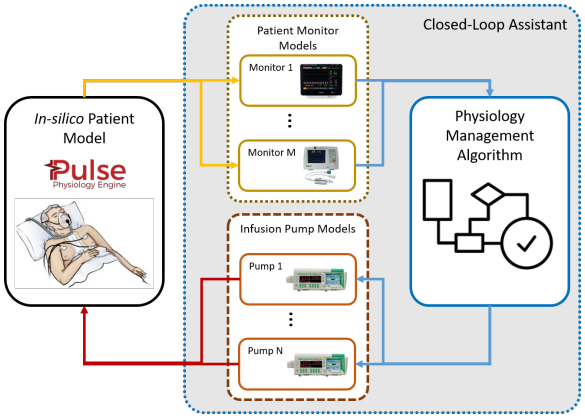


Figure 2: Conceptual picture of framework for evaluating CLAs with *in-silico* patients.

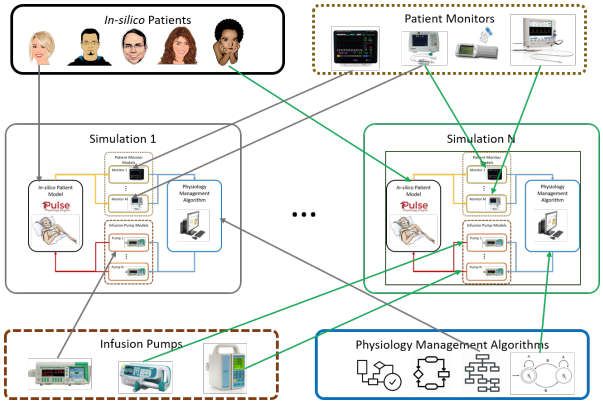


Figure 3: Conceptual picture of design space exploration using *in-silico* patient models.

2.1 Pulse Physiology Platform

The Pulse Physiology platform supports the design, development, and use of physiologic modeling. The Pulse architecture was designed to reduce model development time and increase the usability of the engine in third party application, including software simulations and hardware testing and augmentation, by creating a modular, extensible ontology for simulating the human physiology.

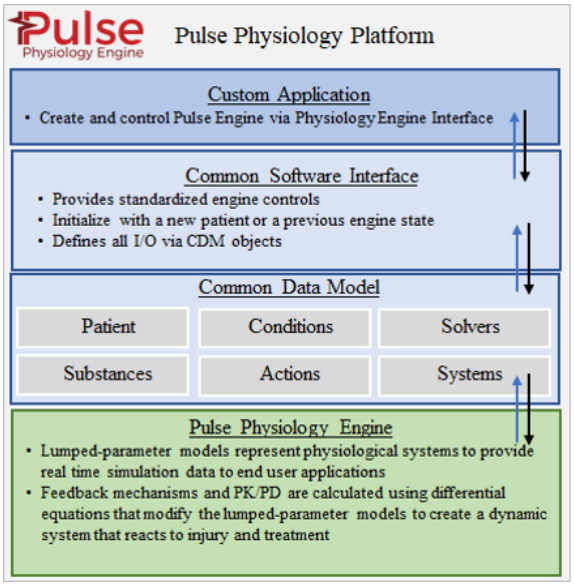


Figure 4: Overview of the Pulse platform

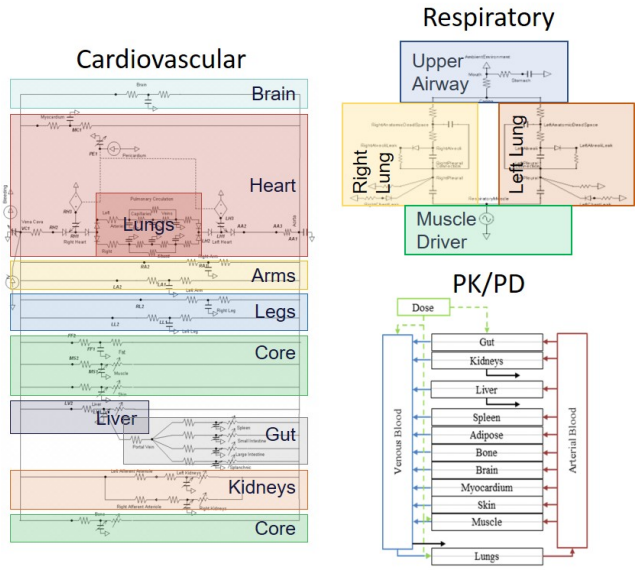


Figure 5: The various models in Pulse.

To accomplish these goals, the Pulse platform includes the Pulse Physiology Engine, a Common Data Model (CDM), and a Software Framework, as shown in Figure 4. This framework has been used to power medical training simulations for a variety of applications [6, 19].

2.1.1 Physiology Engine. The Pulse Physiology Engine is comprised of computation physiology models that represent the different systems of the body, the feedback mechanisms and interactions between the systems, pharmacokinetics/pharmacodynamics (PK/PD), and medical equipment, such as an anesthesia machine.

The different systems of the body have numerical models comprised of lumped-parameters models, which use circuit analogues (e.g. resistors and capacitors) to estimate the behavior of a region of interest/system of the body. Feedback mechanisms required to model trauma and treatment, such as baroreflex responses, are modeled using differential equations that use vital signs as the input to calculate a response, which is then applied to the lumped-parameter components to mimic feedback in the human body.

The PK model represents the migration of substances, specifically drugs through the body, via diffusion and clearance focusing on the plasma concentration over time. After the plasma concentration is accurately calculated, the PD effects, or the effects of the drugs on the vital signs and are implemented. This model is similar to the feedback mechanisms, as the plasma concentration increases, the effect on individual parameters, such as blood pressure, heart rate, and respiratory rate are affected by modifying the lumped-parameter model [8].

The mechanisms in Pulse that connect the various systems are critical to modeling the effects of multiple interventions (*e.g.* combination of ventilation and drug infusions), as well as the side effects of interventions (*e.g.* increased heart rate for an intervention that only reduces blood pressure). (See https://physiology.kitware.com/_system_methodology.html for more information on interaction between various systems in Pulse to account for body feedback mechanisms and multiple interventions).

The Pulse models currently executes with a fixed simulation timestep of 20ms.

For added flexibility, patient variability is a feature of the Pulse Physiology Engine with eight patients included in the open source repository. These patients are defined using a basic set of vital sign parameters that can be adjusted via the provided patient files, including height, weight, heart rate, blood pressure, respiratory rate, alveoli surface area, etc. These parameters are used to initialize the computational physiology models and define a "healthy" patient. After this stabilization occurs, the patient can be given chronic conditions, such as chronic obstructive pulmonary disease and renal stenosis. This process allows for a variety of patients of varying health to be analyzed in conjunction with different disease, injury, and treatment states. In addition to the provided patients, new patients can easily be created using the provided format, parameter definitions, and chronic conditions. More details about patient specification are available on the Pulse website [13].

2.1.2 Command Data Model and Common Software Framework. The Pulse architecture was specifically designed to reduce model development time and increase the usability of the engine in simulations by creating a modular, extensible software system for human physiology. To accomplish these goals, the Pulse architecture includes a Common Data Model (CDM) and a Common Software Framework.

The CDM is an implementation agnostic dictionary-like specification of the data and relationships associated with physiology simulation software systems. The Common Software Framework was designed to reduce development time by providing a place to implement common reusable algorithms to ensure consistent fundamental functionality between programs. Implementation of these

common algorithms provides reusable, validated, supported algorithms that reduce model development time and provide consistent results.

A single interface defines all methods, in terms of the CDM, for both inputs and outputs, as well as controlling the physiology engine. This interface allows software and hardware system developers to integrate Pulse into their hardware and software solutions. It provides a standardized approach for providing instructions to Pulse. Controls are provided to advance time, inject action messages into the engine, and output computed data values for use by sensors and/or display.

2.1.3 Implementation and Validation. The development of the Pulse Physiology Platform follows a high quality software process. This includes providing an open source repository with public version control, a build management system, and a verification and validation suite. The repository is available to the public with an Apache 2.0 license. The platform is built for cross-platform development, including Mac, Windows and Linux, which allows for easy integration with third party hardware and software development.

Ensuring model accuracy and stability is key to the Pulse community. The test scenarios used to ensure model consistency are included in the repository and can be executed after software modifications to ensure the results are correct. These scenarios are built from the validation standards specified by the Pulse team. This standard states that model output is quantitatively validated using data from the literature and the error is reported using a good (<10%), fair (10-30%), poor (>30%) scale. The PK/PD model is also validated for both the plasma concentration curves over time and the vital sign effects. All validation sources and results are reported on the Pulse website (see https://physiology.kitware.com/_cardiovascular_methodology.html).

2.2 Closed-Loop Assistant Components

As mentioned previously, the CLA components consist of the medical devices and the algorithms (assumed to be on a separate computer) that interact with them. Currently, devices consist of abstract models of patient monitors and pumps. The device components are software pieces that take advantage of Pulse's standard interfaces to *directly* observe and effect action on patient state. The algorithm components are software components that interact with the device components to *indirectly* observe and effect action on patient state.

2.2.1 Patient Monitors. Patient monitors take inputs from the patient model and hence interact directly with the Pulse physiology engine. Patient monitors can query any of the available variables from Pulse at a rate whose equivalent sample period is a multiple of Pulse's timestep of 20ms (*i.e.* $50/N$ Hz, $N = \{1, 2, \ldots\}$). The patient monitor can output data to algorithms at a rate whose equivalent period is an integer multiple of its input sample period (*i.e.* rateOfSampleFromPulse/N Hz, $N = \{1, 2, \ldots\}$). Different variables can have different sample and output rates so long as there is a valid relationship between the rate at which a variable is sampled from Pulse and its output rate.

The variables the patient monitor outputs could just be the values from Pulse passed on to the algorithms or other computed values not inherently available from the Pulse model, especially if the computed variable requires multiple sequential samples from the variables directly available from Pulse. For example, although Pulse provides the mean arterial pressure (MAP) directly, we could model a monitor that computes this from the systolic and diastolic blood pressure values from Pulse, or for value like stroke value variation, we could compute this a sequence of multiple stroke volume samples from Pulse.

We can model inaccuracies in values by adding deviations from the true values before they are output to the management algorithms. The current implementation does not support modeling inaccuracies, but is easily extensible to support inaccuracies (which should be in future revisions) and other behaviors (as we develop more use cases).

2.2.2 Physiology Management Algorithms. Physiology management algorithms take inputs from the patient monitors. The algorithms can query any of the available variables from patient monitor at a rate whose equivalent sample period is a multiple of monitors output rate for that variable (i.e. OutputRateFromMonitor/N Hz, $N = \{1, 2, \ldots\}$). Each time new data arrives, the algorithm can choose to make a decision based on inputs so far, or wait till enough input samples are available. Depending on the decision, the algorithm can instruct pumps to take specific actions. The algorithm can make decisions in-between data arrivals (for example, when decision making is triggered by a timeout condition) since it has the opportunity to act in each time step of the simulation.

We currently do not model any issues of the algorithms executing on a computational platform since the case studies we are working with operate at coarse (on the order of minutes) time scales. As we encounter cases with finer time scales (on the order of seconds or less), we will include reasonable issues like computation time causing delayed decision-making and interventions.

2.2.3 Pumps. A pump has substances it can infuse at certain rates. Pumps receive commands to start, adjust, or stop infusions. In general, pumps have a delay parameter that models computation time and time for mechanical components to adjust to a command from the algorithm. This delay can be set to zero to model an ideal scenario without physical effects. Inaccuracies can also be modeled by adjusting the rates before the infusion is applied or even during the application of infusion between adjustments. The current implementation of pumps (and associated configuration parameters) supports only delays, but is easily-extensible to support inaccuracies and other behaviors.

2.2.4 Network. We envision CLAs to be composed using medical application platforms that follow architectures like the Integrated Clinical Environment (ICE) [5]. Hence it is possible that network effects like delays and data losses can occur. We currently do not have an implementation of a network so we assume ideal communication. In future revisions, we intend to implement a network model through which the various CLA components communicate with each other.

2.3 Execution Components and Semantics

Because execution of the whole framework is tied to the execution of Pulse, the framework executes according to a synchronous reactive model of computation [9], where the whole system advances in

discrete timesteps each representing 20ms of wall clock time. At the framework level, there are no zero-delay feedback loops (although these may exist within Pulse itself).

The framework has a simulation engine component where scenario events can be controlled (i.e. actions that happen to the patient independent of the CLA). In this engine, we can decide how the simulation is to be executed. Our current execution cycle proceeds as follows after initialization of all other components:

(1) Check for scenario events (including stop conditions)
(2) Set up and apply any scenario events to patient
(3) Execute one time step of Pulse
(4) Execute patient monitors
(5) Execute algorithms
(6) Execute pumps

Each CLA component has an update function that runs in each time step and determines whether the component reacts to inputs at that point in time or not. We provide a configuration file structure that allows a user to specify properties of devices like input and output rates and accuracies, as well as simulation properties like stop conditions. The rate information from this configuration file is used to determine when devices take in inputs, react to them, and produce outputs. This approach can be used to explore timing and accuracy properties of CLA components without recompiling code (we discuss how simulations are developed below).

2.4 Simulation Development

Currently, developing a simulation in our framework requires writing C++ code and putting information in a configuration file. We provide methods for CLA components (patient monitors, pumps, and algorithms) which handle loading information from the configuration files, basic error checking (whether input and output rates align), and mechanisms for communication between components (monitor-to-algorithm and algorithm-to-pump—we do not support algorithm-to-algorithm communication yet, but intend to have this soon). Currently, the code for the case study simulations for this paper are provided as a demo that a user can work with as a starting point for their simulation. It covers the basic CLA setup and interactions of between CLA components and between the CLA and a patient. In future revisions, we will provide more templates/examples that users can start with and more support code for developing simulations.

Although Pulse supports Windows, Mac, and Linux, our framework is currently being developed on a Linux platform, mostly because the CLA we hope to connect it to for system-in-the-loop simulations relies on Linux features. However, because the software simulation framework is mostly C++ code, we do not see any impediment to using this framework on the other platforms that Pulse already supports.

3 CASE STUDY AND RESULTS

To illustrate the capabilities of the framework and the kinds of explorations we hope to enable for others, we ran a case study simulation based on hypotension management in the ICU based on a conversation with Dr. John McIlwaine, DO, Medical Director for the Center for Telehealth and eICU at the Geisinger Health System about their specific procedures for hypotension management. Most of what we describe here is specific to Geisinger, but is in line with general practice at other institutions. The case study is intended more to be illustrative, so we have simplified many things about the way this is actually done in practice.

3.1 Clinical Scenario and General Strategy for Intervention

A common case of patients in the ICU, especially those who have just come out of surgery, is hypotension (low blood pressure), usually categorized as a mean arterial blood pressure (MAP) below 65mmHg. Maintaining a good MAP is critical to proper perfusion (oxygen delivery to organs). One of the strategies used to manage hypotension is to infuse both fluid (usually saline) and a vasopressor drug (which constricts the blood vessels) like norepinephrine (usually when the patient already has a lower heart rate in addition to hypotension since norepinephrine also increases heart rate). Typically, in addition to this intervention, the clinicians try to determine and address any underlying cause of the hypotension.

Our case study is focused on the intervention using norepinephrine infusion. The general approach to the infusion protocol is shown in Figure 6 in quasi-hybrid system form. The main goal of hypotension management is to get the MAP back into an acceptable range (usually above 65mmHg but below around 85mmHg) and maintain MAP in this range.

There are three major phases to the protocol. The first is the initial reaction to the hypotension where the patient is hit with the maximum allowed dose of the drug and fluid infusion is also started (not shown in Figure 6 because fluid infusion, at least in this case study, remains at a constant rate). After the MAP has increased initially to a threshold (INC in Figure 6) usually around 100mmHg.

The second phase begins where the dose is reduced (to some percentage of the max dose) to prevent issues like hypertension or other side effects from having a high does of the drug. If the MAP continues to rise, the dose is reduced until the drop is seen. If the MAP begins to drop in response to the reduce dose, then the third phase begins where the dose is adjusted to maintain the MAP in the target range (MAP_MIN and MAP_MAX in Figure 6, usually 65mmHg and 85mmHg respectively). The does is increased when the MAP falls below range and and is reduced when the MAP goes above range. This phase continues until it is determined that infusion of norepinephrine is no longer required. In some cases, this can go on for 24 hours or more. The checks for MAP between actions usually happens at roughly 10 minute intervals, though when the patient is below range, the MAP is checked more frequently about every 5 minutes.

3.2 Simulation Setup

We used the framework to examine two slightly different strategies for the infusion protocol across a small patient population. This allows us to illustrate some variability both in the the CLA and in the patients it must interact with.

For each patient, we mimicked a hypotensive patient coming out surgery by starting them off with a lower baseline MAP and hemorrhaging the patient till the MAP dropped to 70mmHg, at which point we started the protocol. We ran the whole scenario for 60 minutes of simulation time for each patient.

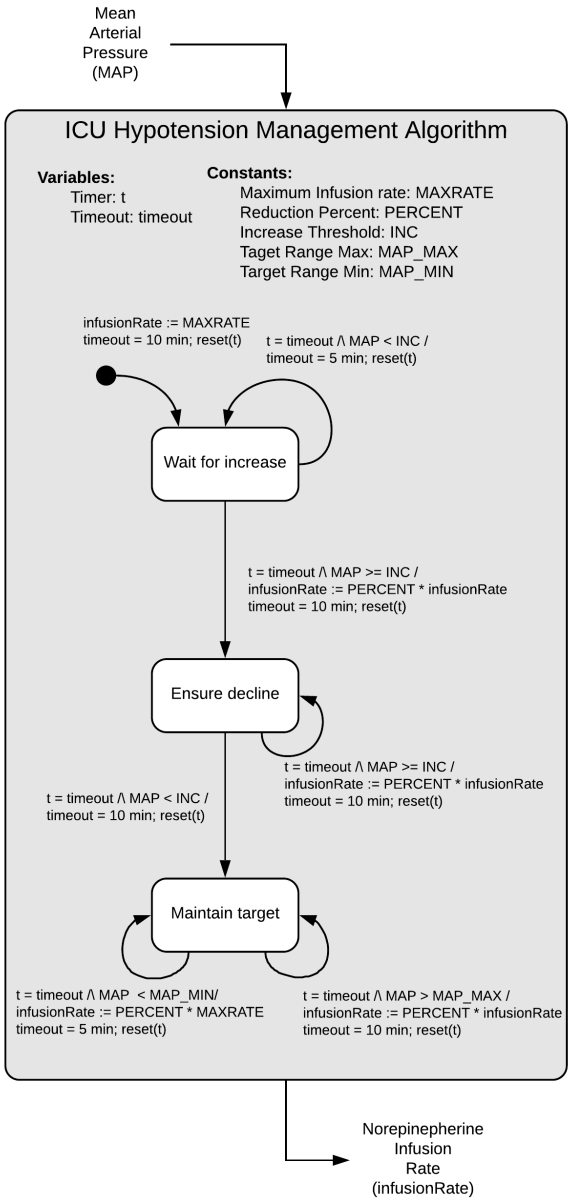


Figure 6: The norepinephrine infusion protocol.

To get the two different strategies we parameterized the infusion protocol described in the previous section (illustrated in Figure 6), using the constants in that model as the parameters. The only variable in we changed between both algorithms was the percentage reduction (and associated increase) in dose. The full infusion protocol parameters for both strategies are given below.

The patient monitor is modeled as one that produces MAP directly. For simplicity, we pull this value directly from Pulse rather can calculate it from the systolic and diastolic pressures, which

Table 1: Simulation Parameters

Parameter	Symbol in Figure 6	Units	Value	
			Algorithm 1	Algorithm 2
Patient MAP at start	-	mmHg	70	
Saline Rate	-	ml/hr	50	
Norepinephrine Concentration	-	ug/ml	16	
Maximum Norepinephrine Infusion Rate	MAXRATE	ml/hr	84	84
Percentage Reduction in Norepinephrine Infusion Rate	PERCENT	%	75	87.5
MAP Threshold to start reducing dose	INC	mmHg	80	80
Upper Threshold of MAP Target Range	MAP_MAX	mmHg	80	80
Upper Threshold of MAP Target Range	MAP_MIN	mmHg	75	75

is what a real device would do. We also assume it is accurate and do not introduce any error. The patient monitor 'computes' MAP 1.2 times per minute and can output this value 0.6 time per minutes. Both versions of the algorithm ask for this value 0.6 times per minute, even though the current value is used either very 10 or 5 minutes. The pumps (for saline and norepinephrine) are modeled without any inaccuracies but with a delay of 15 seconds.

3.3 Results

Figure 7 shows the result of running the algorithm on of our patients (Hassan). It illustrates the different kinds of information we can glean from the simulation. First, we see the 'view of the world' from the perspective of each entity in the simulation. The patient data shown are the 'ground truth' physiologic values generated by Pulse. The patient monitor values are the vales that the monitor reports as what it observed after interacting with the patient to obtain them. These could be different from the ground truth values for various reasons. Presenting the results this way foregrounds the fact that in a real scenario the monitor goes through a sensing process where physical signals are converted to digital values, and in many cases with additional digital processing performed to decide on the final value the monitor will report as the observed value.

The algorithm then sees a further sampled version of what the monitor observes. This shows that although a monitor can observe values at higher time resolutions, it may only report values to other systems at lower time resolutions so the algorithm's view of the patient may not necessarily be the same as the monitor's view of the patient. As mentioned previously the monitor samples MAP values at rate of 1.2/min, and the algorithm sees values at 0.6/min.

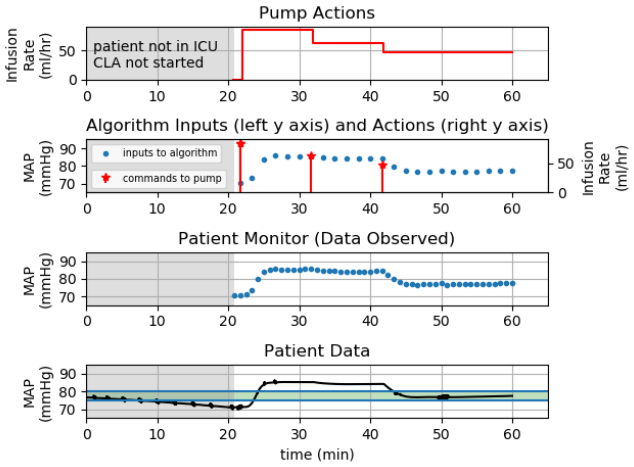


Figure 7: Results from CLA running algorithm 1 interacting with a single patient.

Based on the values the algorithm receives, it can send commands to the pump to adjust infusion rates. The pump has a delay from when it gets command till when the new infusion rate starts (not discernible on the figure because it is 15s in this simulation).

In Figure 7 we have highlighted the target MAP range in the patient data plot. We have also highlighted the period before the CLA starts interacting with the patient in all the plots. This allows us to see different parts of the scenario more clearly. Though not shown, one could also highlight what phase of the protocol the algorithm is in order to check that the algorithm is in the right phase given the past and current events.

For the case shown in Figure 7, the algorithm is able to get Hassan out of the hypotensive state in the first few minutes and then into the target range within about 25 minutes from the start of its operation. We can also verify that the algorithm is indeed following the logic in Figure 6 according to the parameters in Table 1 by looking at the MAP data and the algorithm's reactions.

Figure 8 shows the results of running the two different algorithms on three patients (two males, named Hassan and Gus, and one female named Diana). We used three out of the available patients in Pulse because of a limitation with its homeostatic feedback and PD model that only allows these three patients to respond to the initial drop in MAP (to simulate coming out of surgery) and to infusion of norepinephrine the way we expect.

Nevertheless, the results are meant to be illustrative and even with these three patients we see some variation in outcomes. We have only shown the ground truth patient data and the CLA's reaction to the data since we are interested in how the algorithms perform overall. (Note that the patient data also represents the patient's reaction to the CLA's actions). For each case, we can dig deeper like in Figure 7 to see how the specific behavior of the various entities may be contributing to the outcomes we observe.

Both algorithms have the same initial effect on patients because they both administer norepinephrine at the maximum dose when they start. Diana seems to react much quicker to the action of the drug. Algorithm 1 gets two of the patients (Hassan and Gus) into the target range within the simulation time since it reduces the dose each by a higher amount than algorithm 2 each time reduction is needed. Algorithm 2 struggles to get any of the patients into the target range. Though algorithm 1 also struggles to get Diana into the target range even after reducing the dose, it starts to lower Diana's MAP after the initial increase closer to the end of the simulation.

These results illustrate some the insights we hope users of the framework can gain about their system. We see the general difference between the algorithms (one is generally faster at getting patients into the target range than the other) and the common issues across the algorithms (there is one patient both struggle with).

4 RELATED WORK

4.1 Multi-Purpose Physiology Engines

Computational physiology models, often termed physiology engines, are available both commercially and as open-source toolboxes for integration with external software and hardware [1, 7, 10]. These engines vary in their usability by cost, service and support availability, and model and validation documentation. Many of the commercial physiology engines are expensive and provide limited support for those hoping to integrate the models into their proprietary products. The BioGears engine [1] is an open source physiology engine developed as a Department of Defense program. However, it has some licensing concerns with commercial, proprietary applications and a lack of current development and support that makes it difficult to integrate with external applications. The Pulse Physiology Engine is a fork of the BioGears engine that has been actively developed by experts in open source software development and support. Pulse extended BioGears to create an open source repository with a permissive Apache 2.0 license that is multi-platform (Windows, Mac, and Linux).

4.2 Frameworks for Evaluating Closed-Loop MCPS

Two of the well-known frameworks for evaluation of closed-loop MCPS are the simulation platform for the artificial pancreas [17] and the that for pacemakers [24]. Both of these frameworks consist of in-silico patient models as well as models of the MCPS. In the artificial pancreas case, for the MCPS there are models of continuous glucose monitors (including their inaccuracies and delays), models of the management algorithms and simple pump models. The *in-silico* patient population contains 300 patients (100 each of adults, adolescents, and children), and that population and the results from simulations with them are currently accepted by the U.S. Food and Drug Administration (FDA) as an alternative to animal trials, which speeds up the concept-to-human-trial time, and reduces the cost of evaluating new concepts.

The pacemaker work, though it does not yet enjoy the same status as the artificial pancreas work with regards to use in the regulatory process still provides an approach that is worth building on. In particular, unlike the artificial pancreas work, the pacemaker work provides system-in-loop simulation capabilities with *in-silico* patients running on hardware that can interact with real

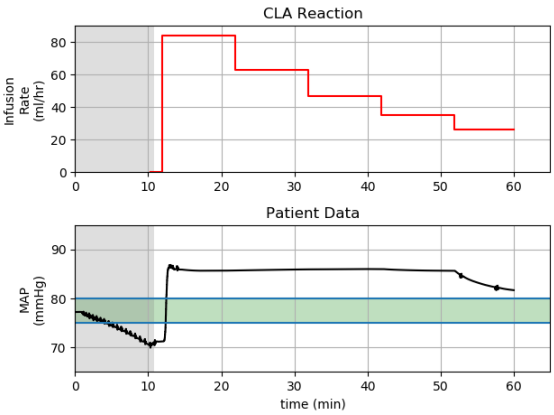
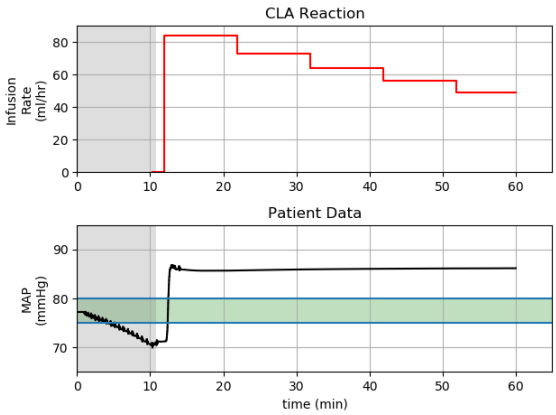
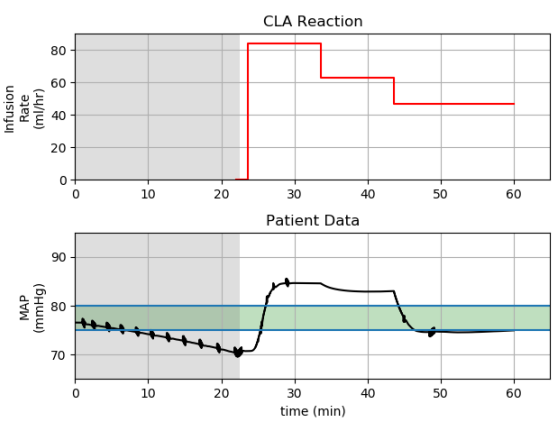
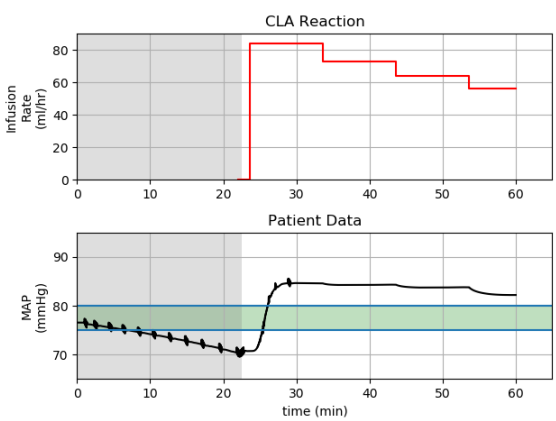
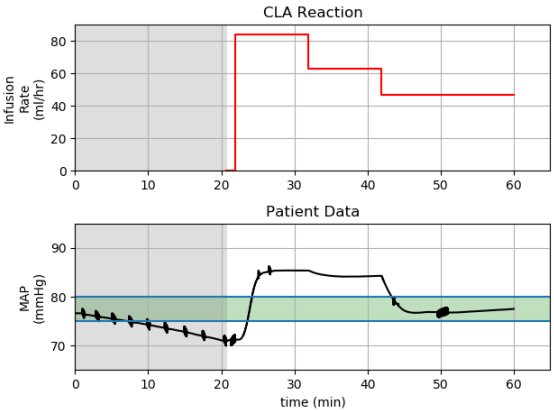
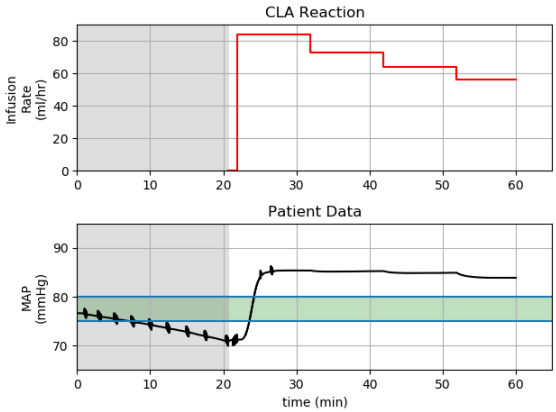
Patient	Algorithm 1 Outcome	Algorithm 2 Outcome
Diana		
Gus		
Hassan		

Figure 8: Results from *in-silico* trial of both algorithms on a small patient population.

implantable cardiac devices [12]. It also has complementary formal-methods-based tools for design evaluation. These are critical to the design process for MCPS [15].

The work presented here for CLAs builds on these efforts, extending these ideas to MCPS for physiology management in perioperative and critical care. Though simulation-based approaches have been explored in these contexts (e.g. work on safety of patient-controlled analgesia [16] and application of the artificial pancreas framework to glucose management in critical care patients [18]), we are not aware of a general purpose framework that allows consideration of many different scenarios and variables.

5 CONCLUSION AND FUTURE WORK

Leveraging connectivity capabilities of medical devices to enhance approaches to patient care, especially though the use of closed-loop technologies, is gaining interest within the medical community. For example, the most recent annual meeting of the Society for Technology in Anesthesia, a mix group of medical professionals and medical technologist, had a whole session dedicated to automated drug delivery and closed loop systems [21]. With the work to make open versions of CLA technologies available to the community [22], including our own efforts [4], many will soon be able to experiment with these ideas and develop systems for particular contexts. The ability to evaluate ideas early in the design process in cost-effective ways is critical to realizing these technologies. The work presented in this paper provides an approach that can help with this. In particular, because both the patient physiology models and the CLA models are available to the designers and their medical domain experts, both can collaborate to understand how different physiology factors affect efficacy of design, or conversely how a particular design may interact with a particular physiology.

We are continuing to improve the Pulse physiology engine, particularly for this context where patient physiology has been altered by prolonged period of trauma or by going through surgery, and for cases of continuous infusion as an intervention over long periods of time. We are also working on a system-in-the-loop simulation extension to this work. As mentioned previously, we have developed our own CLA experimentation platform. We also have access to computer-controllable hardware that can simulate the physical signals that patient monitors observe from patients. We are currently working on getting Pulse to drive the hardware and receive actions from the infusion pump in real-time. With this in place, we can begin to experiment with CLA in real time to gain insights into some of the issues that arise in more realistic contexts.

Taking inspiration from the pacemaker work, we also interested in developing a complementary formal-methods-based framework for design and evaluation of CLAs. The Pulse model is essentially a discrete time hybrid system (though a large and complex one). The CLA models can also be viewed this way. With some simplifications, we could develop formal specifications for the goals of the CLA (using ideas from [3] to relate these to patient outcomes) and formal verify candidate CLA designs. Additionally, we could use these specifications to generate candidate designs using controller synthesis techniques (e.g. [20]). In both cases, we could then run higher-fidelity in-silico trials (software only or system-in-the-loop) to further evaluate the candidate designs. This complementary formal-methods-based framework enable faster design-space explorations by helping prune the design space and reducing the number of high fidelity simulations that need to be performed.

ACKNOWLEDGMENTS

The authors would like to thank Dr. John Mcilwaine for taking time to explain the infusion protocol and expected physiology response to us.

REFERENCES

[1] Applied Research Associates, Inc. 2017. BioGears. https://www.biogearsengine.com/. (2017).
[2] I. Armenti, P. Asare, J. Su, and J. Lach. 2012. A methodology for developing quality of information metrics for body sensor design. In *Proceedings - Wireless Health 2012, WH 2012*. https://doi.org/10.1145/2448096.2448098
[3] Philip Asare. 2015. *A Framework for Reasoning about Patient Safety of Emerging Computer-Based Medical Technologies*. Ph.D. Dissertation. University of Virginia, Charlottesville, VA. https://doi.org/10.18130/V3WG3B
[4] Philip Asare, Mahmood Arifin Chowdhury, Taimoore Rajah, S. Mark Poler, Mohammed Shah, Peter Guion, Jeffrey Martin, Kevin Driscoll, Qianhong Wu, Andrew Mannes, and C. Nataraj. 2016. A system for semi-automated management of blood loss during surgery: Preliminary results. In *2016 IEEE Healthcare Innovation Point-Of-Care Technologies Conference (HI-POCT)*. IEEE, 216–219. https://doi.org/10.1109/HIC.2016.7797735
[5] ASTM International. 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 Mode (ASTM F2761-2009). (2009).
[6] Randy Brown, Steve McIlwain, Brad Willson, and Matthew Hackett. 2016. Enhancing Combat Medic training with 3D virtual environments. In *2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH)*. IEEE, 1–7. https://doi.org/10.1109/SeGAH.2016.7586266
[7] CAE Healthcare. 2018. Patient Simulation Products. https://caehealthcare.com/patient-simulation. (2018).
[8] Rachel B. Clipp, Aaron Bray, Rodney Metoyer, M. Cameron Thames, and Jeffrey B. Webb. 2016. Pharmacokinetic and pharmacodynamic modeling in BioGears. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 1467–1470. https://doi.org/10.1109/EMBC.2016.7590986
[9] Stephen A. Edwards and Edward A. Lee. 2003. The semantics and execution of a synchronous block-diagram language. *Science of Computer Programming* 48, 1 (jul 2003), 21–42. https://doi.org/10.1016/S0167-6423(02)00096-5
[10] HC Simulation, LLC. 2016. Hummod. http://hummod.org/. (2016).
[11] Insup Lee, G.J. Pappas, Rance Cleaveland, John Hatcliff, B.H. Krogh, Peter Lee, Harvey Rubin, and Lui Sha. 2006. High-Confidence Medical Device Software and Systems. *Computer* 39, 4 (apr 2006), 33–38. https://doi.org/10.1109/MC.2006.127
[12] Zhihao Jiang, Miroslav Pajic, Allison Connolly, Sanjay Dixit, and Rahul Mangharam. 2010. Real-Time Heart Model for Implantable Cardiac Device Validation and Verification. In *2010 22nd Euromicro Conference on Real-Time Systems*. IEEE, 239–248. https://doi.org/10.1109/ECRTS.2010.36
[13] Kitware Inc. 2016. Pulse Physiology Engine. https://physiology.kitware.com/. (2016).
[14] Boris P Kovatchev, Marc Breton, Chiara Dalla Man, and Claudio Cobelli. 2009. In Silico Preclinical Trials: A Proof of Concept in Closed-Loop Control of Type 1 Diabetes. *Journal of Diabetes Science and Technology* 3, 1 (jan 2009), 44–55. https://doi.org/10.1177/193229680900300106
[15] Miroslav Pajic, Zhihao Jiang, Insup Lee, Oleg Sokolsky, and Rahul Mangharam. 2012. From Verification to Implementation: A Model Translation Tool and a Pacemaker Case Study. In *2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*. IEEE, 173–184. https://doi.org/10.1109/RTAS.2012.25
[16] Miroslav Pajic, Rahul Mangharam, Oleg Sokolsky, David Arney, Julian Goldman, and Insup Lee. 2014. Model-Driven Safety Analysis of Closed-Loop Medical Systems. *IEEE Transactions on Industrial Informatics* 10, 1 (feb 2014), 3–16. https://doi.org/10.1109/TII.2012.2226594
[17] Stephen D Patek, B Wayne Bequette, Marc Breton, Bruce A Buckingham, Eyal Dassau, Francis J Doyle, John Lum, Lalo Magni, and Howard Zisser. 2009. In Silico Preclinical Trials: Methodology and Engineering Guide to Closed-Loop Control in Type 1 Diabetes Mellitus. *Journal of Diabetes Science and Technology* 3, 2 (mar 2009), 269–282. https://doi.org/10.1177/193229680900300207
[18] Stephen D Patek, E Andy Ortiz, Leon S. Farhy, Jennifer Mason Lobo, James Isbell, Jennifer L Kirby, and Anthony McCall. 2015. Population-specific models of glycemic control in intensive care: Towards a simulation-based methodology for protocol optimization. In *2015 American Control Conference (ACC)*. IEEE, 5084–5090. https://doi.org/10.1109/ACC.2015.7172132

[19] Lucas Potter, Sreekanth Arikatla, Aaron Bray, Jeff Webb, and Andinet Enquobahrie. 2017. Physiology informed virtual surgical planning: a case study with a virtual airway surgical planner and BioGears, Robert J. Webster and Baowei Fei (Eds.). 101351T. https://doi.org/10.1117/12.2252510

[20] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M. Murray, and Sanjit A. Seshia. 2015. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems Computation and Control - HSCC '15*. ACM Press, New York, New York, USA, 239–248. https://doi.org/10.1145/2728606.2728628

[21] Society for Technology in Anesthesia. 2017. Annual Meeting 2018 Brochure. https://www.stahq.org/userfiles/files/STA_18AM_Reg_Bro%28347%29.pdf. (2017).

[22] The Medical Device "Plug-and-Play" Interoperability Program. 2015. OpenICE. https://www.openice.info/. (2015).

[23] Zhihao Jiang, A Connolly, and R Mangharam. 2010. Using the Virtual Heart Model to validate the mode-switch pacemaker operation. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, 6690–6693. https://doi.org/10.1109/IEMBS.2010.5626262

[24] Zhihao Jiang, M Pajic, and R Mangharam. 2012. CyberâĂŞPhysical Modeling of Implantable Cardiac Medical Devices. *Proc. IEEE* 100, 1 (jan 2012), 122–137. https://doi.org/10.1109/JPROC.2011.2161241