Synthesis of Context-Aware Safety Monitors for Medical Cyber-Physical Systems: A Hybrid Model and Data Driven Approach

***Abstract*—Rapid advances in sensing and computing technology have led to the proliferation of medical Cyber-Physical Systems (CPS) in personalized and clinical settings. However, the increasing device complexity, shrinking technology sizes, and shorter time to market have resulted in major challenges in ensuring the reliability, safety, and security of medical devices. This paper presents a hybrid model- and data-driven approach for the synthesis of safety monitors that can continuously detect faults and mitigate hazards in medical CPS (MCPS). The synthesis process starts with the identification of safety requirements using systems-theoretic (STPA) hazard analysis and their translation to Linear Temporal Logic (LTL) properties for the detection of unsafe system context. The extracted LTL properties are further refined using the data collected from the closed-loop simulation of the controller with a dynamic patient model. The final synthesized monitor is integrated with the target control software as a wrapper that only has access to the input-output interface (sensor and actuator values) and performs real-time execution of LTL rules for preemptive detection of unsafe controller commands *before* they are delivered to the actuators and cause harm to patient. We demonstrate the effectiveness of our approach in simulation using a closed-loop artificial pancreas system (APS). The results shows a safety monitor developed with this approach demonstrates up to 15.6% increase in average detection accuracy (as measured by the F_1 score) and 50.6% reduction in the number of false-positive (FP) alarms, and a slightly longer (1%) reaction time (time between detection and actual occurrence of hazard) compared to a baseline.**

***Index Terms*—safety, resilience, run-time verification, online monitoring, error detection, hazard analysis, medical.**

I. Introduction

Medical Cyber-Physical Systems (MCPS) are increasingly deployed in various safety-critical diagnostic and therapeutic applications. These devices often need to process critical clinical information or deliver direct treatment to patients while satisfying tight timing, power, and area constraints. However, the increasing device complexity, shrinking technology sizes, and shorter time to market (TTM) have resulted in major challenges in ensuring the reliability, safety, and security of medical devices. Recent studies have shown the susceptibility of medical devices, such as patient monitors, infusion pumps, and implantable pacemakers, to accidental faults or malicious attacks with potential adverse impact on patients [1]–[5].

Analysis of past data on recalls and adverse events involving medical devices shows that the inclusion of run-time safety and security monitoring and fault-tolerance mechanisms might be ignored or overlooked in the design of these devices due to the limited TTM and timing and resource constraints [1]. Some of the existing monitoring techniques for MCPS rely on fixed rules and general medical guidelines and often ignore the underlying context, including dynamic systems and patient status [4]–[6]. On the other hand, many previous works focused on the design and synthesis of run-time verification and online monitoring techniques using safety properties specified as Finite State Machines (FSM) or LTL [7]–[11]. Most recent and relevant works in this area focus on robust online monitoring in resource-constrained CPS by learning LTL properties from system traces [12]. However, most of the previous works rely on ad-hoc specification of safety properties without considering the control system interactions and context in CPS. Systems Theoretic Process Analysis (STPA) [13] is a hazard analysis technique driven by systems and control theories that overcomes this limitation by modeling the accidents as complex dynamic processes resulting from inadequate control mechanisms that violate safety constraints. Using STPA, the safety properties are defined by identifying the underlying system context that could potentially lead to unsafe control actions and safety hazards. But despite improvements over traditional techniques, STPA still lacks a solid formal framework for specifying unsafe system context [14] [15], and there is a gap between the high-level safety requirements identified from STPA and the low-level formal specification of safety properties that can be used for online monitoring. Further, most of the existing online monitoring solutions do not provide the ability for *early* detection of safety property violations and prevention of hazards.

In this paper, we propose a system-resilience design methodology in which context-aware safety monitors can be automatically synthesized from the safety requirements identified through a control-theoretic hazard analysis process and be integrated with the control software of a target MCPS at design-time to continuously detect and mitigate the delivery of unsafe control commands to the patient at run-time. This approach is based on the high-level safety context table of the target MCPS developed using STPA, which overcomes the limitations of current monitoring techniques often designed based on fixed requirements and static rules from the medical domain. Besides, the synthesis of the monitor is done based on only the knowledge of system specification and information available from the hazard analysis process, without having access to the internal design of the controller. Thus, the monitor can be seamlessly integrated as a wrapper with the controller and has the potential of decreasing controller complexity while increasing the diversity between the controller and monitor logic and thus improving the reliability of the overall system. Another potential application of such an independently designed monitoring system could be the certification of a target controller using run-time verification and before actual deployment and use by the patients.

The main contributions of this paper are as follows:

- We propose a system-resilience methodology for MCPS based on synthesis and integration of context-aware safety monitors that consider the underlying system context, including not only patient status but also controller states, to detect early signs of potentially unsafe control commands in the cyber layer and stop their execution in the physical layer before causing harm to patients.
- We present a hybrid model and data-driven approach for the synthesis of the context-aware monitors that bridges the gap between the less formal STPA hazard analysis process with the formal LTL specification of safety properties.
- The proposed synthesis method can be applied to any MCPS by just knowing the functional specification, the input-output interface, and the STPA derived safety requirements for the target MCPS and does not need access to the internals of the device design and implementation.
- Our experimental results on the closed-loop OpenAPS system show that the proposed context-aware monitoring technique outperforms a baseline monitor designed using medical guidelines in timely and accurate detection of impending safety hazards, with up to 15.6% increase in average detection accuracy (F1 score) and 50.6% reduction in the number of false-positive (FP) alarms while providing a slightly longer (1%) reaction time (time between detection and actual occurrence of hazard).

The rest of this paper is organized as follows. Section II outlines relevant background and the motivating examples behind this research undertaking. Section III illustrates both the conceptual definition and detailed deployment procedure of the proposed framework. Section IV and V proposes a concise evaluation and a brief discussion of related works. Section VI provides a conclusion and directions for future research.

II. Background

A. Safety Context in Cyber-Physical Systems

CPS are designed by the tight integration of cyber components and software with the hardware devices and the physical world. As shown in Figure 1, in the human-in-the-loop MCPS (e.g., cardiac pacemakers, artificial pancreas systems (APS), and surgical robots), the controller software often needs to continuously interact with the physical processes (e.g., patient's body) and human operators (e.g., physicians, nurses) in real-time to not only adapt to the constantly changing and uncertain physical environment but also take into account the operator's preferences, intent, and behavior [16].

Safety as an emergent property of CPS is context-dependent and should be controlled by enforcing a set of constraints on the behavior of system and control actions given the current system context [17]. The system context can be formally defined as the set of conditions under which the control actions are issued. Systems-Theoretic Accident Model and Processes (STAMP), which is an accident causality model driven by the concepts in systems and control theories [13], defines the safety context as the combination of state variables used by the controller for distinguishing between safe and unsafe control actions.

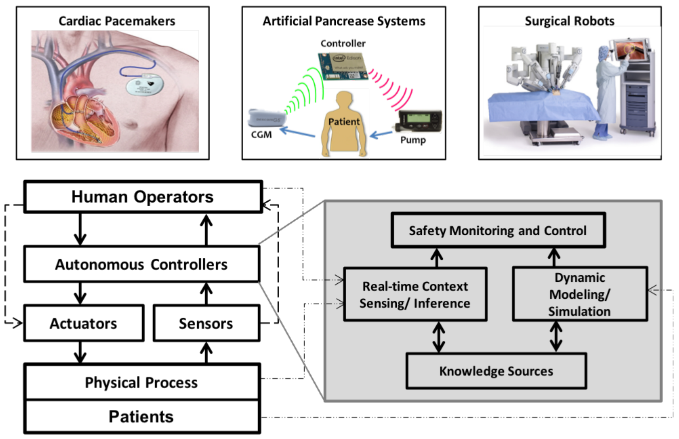


Fig. 1. Generic MCPS and Safety Context

In such systems, safety hazards and incidents might result from unsafe commands of the controller due to errors in input and sensor data, bugs in the control algorithm, faulty controller hardware, or a malfunctioning actuator. A simple monitoring mechanism might involve checking the values of commands defined by the medical guidelines. However, such a monitoring system does not take into account the current status of the system and dynamics of the patient and might incorrectly classify safe commands, leading to a large number of false alarms, or, more importantly, might miss the detection of unsafe control actions. For instance, in Surgical Robot, it is hard to ensure safety based on just the constraint of position and velocity of its arms. Examples of adverse events have been previously reported to the U.S. FDA on medical devices, like adverse events involving patient monitoring systems in hospitals where data errors and algorithmic inadequacies caused false alarms or missed detection, leading to patient harm [18] [19].

B. Case study

As a case study, consider the APS, where a continuous glucose monitor (CGM), acting as the sensor, periodically checks the blood glucose (BG) concentration in the patient's body and transmits it to the controller. The control software estimates the current patient status and calculates the next recommended insulin value that needs to be delivered to the patient's body. This insulin information is sent as a command to the insulin pump, and the pump, acting as the actuator, delivers the insulin to the patient.

An example of adverse events is presented in Figure 2, from which we can see that the baseline monitor generates an alarm when BG value crosses a specific limit, 180 in this case. However, it does not consider the action of the controller

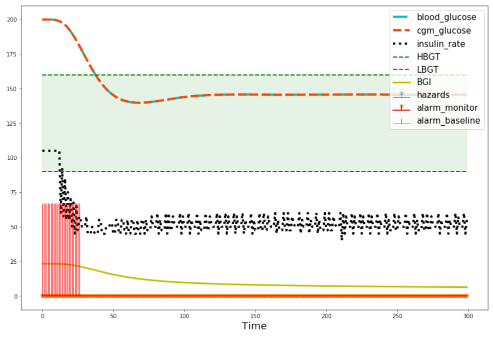


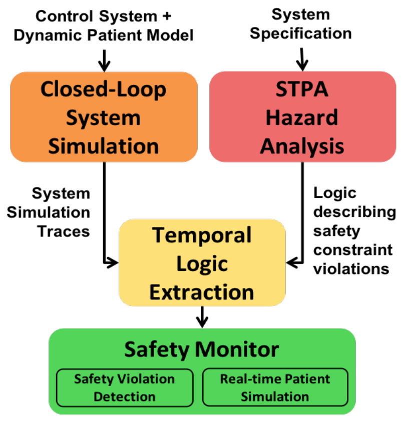
Fig. 2. Monitor does not consider the action of controller

that actively delivers sufficient insulin to bring the glucose in the normal range (shaded region). Thus, the baseline creates a lot of false alarms.

To demonstrate the ideas proposed in this work, we developed a closed-loop APS consisting of the OpenAPS [20] as the controller and the open-source GlucoSym simulator [21] as our patient simulator and model.

III. Synthesis of Context-Aware Safety Monitor

Figure 3 shows the overall steps in our approach for the synthesis of a context-aware safety monitor. We first conduct a rigorous hazard analysis using STPA to identify the potential unsafe scenarios that might lead to safety hazards for the patient. We develop a system context table consisting of all the control actions and different system contexts that might lead to unsafe control actions. The output from this step is the logic describing safety requirements or set of safety constraints that should not be violated by the target controller. Then, by performing a closed-loop system simulation, we generate simulation traces that can help with further refinement of the logic learned from the hazard analysis step. Finally, the logic describing the unsafe system context is translated into linear temporal logic and is provided along with the simulation data to a tool for automated extraction of logic rules for the implementation of the monitor. The final monitor is composed of the extracted logic rules for the detection of safety violations along with a simulated patient model that predicts the outcome of providing a control command or the risk of harming the patient in a near-future before the command is delivered to the patient.



Control System +
Dynamic Patient Model
System
Specification
Closed-Loop System Simulation
STPA Hazard Analysis
System Simulation Traces
Logic describing safety constraint violations
Temporal Logic Extraction
Safety Monitor
Safety Violation Detection
Real-time Patient Simulation

Fig. 3. Steps for Synthesis of Safety Monitor

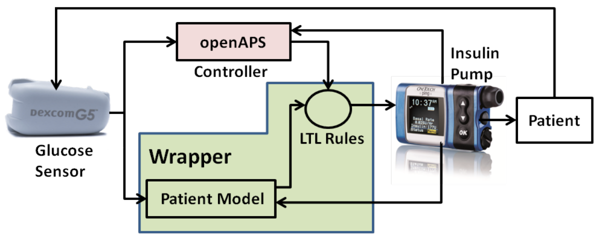


Fig. 4. Context-aware Monitor Integrated as a Wrapper with the OpenAPS Control Software

Figure 4 shows an example of the context-aware monitor synthesized from the STPA context table and integrated as a wrapper with the OpenAPS control software. As shown in the figure, the monitor only has access to the input and output of the target control software and is exclusively designed based on the knowledge of system specification and hazard analysis process.

Model assumption: This paper focuses on the detection of unsafe control actions caused by faults either in the hardware or software of the controller. We assume the sensors and actuators works normally.

A. Systems Theoretic Hazard Analysis

STPA [17] is a hazard analysis technique based on control and systems theories that, unlike traditional risk/hazard analysis techniques, focuses on the dynamic behavior of a system. It is in particular appropriate for hazard analysis in MCPS because (1) it doesn't require access to the internals of device and code, and (2) it models the system as a hierarchical control structure with multiple control loops, including the human operators, automated controllers, and cyber and physical processes. While traditional techniques are based on reliability theory and use component failure as the key point for hazard analysis, STPA is based on systems theory and considers safety as an emergent property of the system and tries to solve it as a control problem. The STPA hazard analysis process starts by identifying the accidents, system hazards associated with those accidents, and the safety requirements (constraints) for the system. Then unsafe control actions in each loop of control structure are identified. An unsafe control action is defined as one of the following four types:

1) A control action required is not provided.
2) An incorrect control action is provided.
3) A control action is provided too early or too late.
4) A control action is stopped too early or applied too long.

Next, the potential causes for unsafe controls are determined by considering any possible flaws in the inputs (sensor measurements), control algorithm or process model, outputs (actuator commands), or feedback received by the controller. More specific steps are as follows.

Accidents: Based on the functionality of the OpenAPS controller, and according to the STPA Analysis [17], we first characterized two types of accidents that might happen to the patients due to faults in the APS controller:

- **A1:** Hypoglycemia— the BG level is too low.
- **A2:** Hyperglycemia— the BG level is too high.

Though hypoglycemia usually happens when the BG is less than 70 mg/dL, there isn't a fixed threshold to different patients. So is the high glucose level of hyperglycemia. To better describe these accidents, we adopted the concept of Risk Index (RI) [22], [23] to define the hazardous situation that indicates a high risk of hyper- or hypo-glycemia in the patient's body.

Hazards: Severe situations indicating accidents, used as the ground truth to evaluate the performance of the proposed monitor.

- **H1:** RI is higher than 5 and keep increasing, which might result in **A1**.
- **H2:** RI is higher than 9 and keep increasing, which may lead to **A2**.

The detail of the RI is described in sections IV-C and **??**.

Safety Control Structure: We modeled the safety control structure of the closed-loop APS as shown in Figure 5. We identified different system conditions in which different control actions might be unsafe and lead to a hazardous situation. First, we defined the state variables and then analyzed all the control actions with different combinations of these state variables in which the control actions are considered as unsafe. For unsafe control actions, the following two specific scenarios were considered:

- a required control action was not provided and
- a control action is provided when not required

In our experiment, the command is delivered to the pump with a frequency of 5 minutes and different duration. So the last two types of unsafe action like too early, late, short or long are also covered in the first two types.

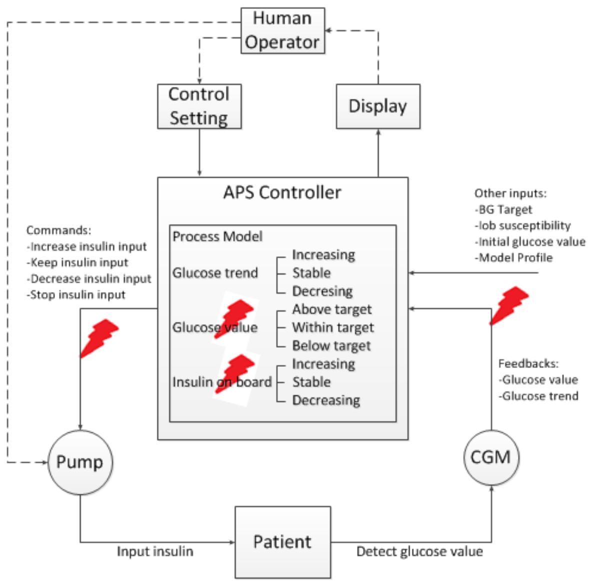


Fig. 5. Control Structure with the OpenAPS Controller's Process Model

STPA Context Table: The construction of a context table starts from selecting the controller, state variables, and control actions from the control structure. The context table developed for the OpenAPS controller is shown in Table I. The first column indicates different control actions provided by the controller, and the next two columns are the state variables BG and Insulin On Board (IOB). The last two columns determine whether the control action leads to a hazardous situation in a specific context if that control action is or isn't provided. *dec_insulin, inc_insulin, zero_insulin, and keep_insulin* refer to decrease, increase, stop, and keep insulin delivery, respectively.

The rows of the context table are populated with a different combination of the control actions and state variables. Each of the combinations is referred to as a context for the system. As an example, the first row includes a specific context where the BG level is greater than the high blood glucose target (HBGT) and rising, with IOB below a certain threshold and also increasing. In this context, providing a *dec_insulin* command might lead to a hazardous situation H2.

Our original context table for OpenAPS included several redundant rows corresponding to similar or not hazardous system context, which we pruned to generate the summarized context table here. For example, about glucose value, we have three situations: rising, stable, and falling. However, when the glucose is stable, no matter how the insulin changes, we will eventually detect whether it is a safe action or not by the other two situations. Therefore it makes the context table clean to remove this redundant rule. In the final context table (Table. I), each row represents a safety property or constraint, that if violated by the controller, might lead to one of the safety hazards identified earlier. Thus, each row of the context table can be used as a logic rule to be checked by a safety monitor, that has access to all the values of state variables and the control commands from the controller, to detect any potential harmful control actions.

TABLE I
STPA CONTEXT TABLE

Rule No.	Control Action	BG	IOB	Hazards? Control Action	
				Provided	Not Provided
1	dec_insulin	>HBGT; rising	falling; < Th1	H2	
2	dec_insulin	>HBGT; rising	stable; < Th2	H2	
3	dec_insulin	>HBGT; falling	rising; < Th3	H2	
4	dec_insulin	>HBGT; falling	falling; < Th4	H2	
5	dec_insulin	>HBGT; falling	stable; < Th5	H2	
6	inc_insulin	< LBGT; falling	rising; > Th6	H1	
7	inc_insulin	< LBGT; falling	falling; > Th7	H1	
8	inc_insulin	< LBGT; falling	stable; > Th8	H1	
9	zero_insulin	>HBGT	< Th9	H2	
10	zero_insulin	< Th21	*		H1
11	keep_insulin	> HBGT; rising	not rising; <Th10	H2	
12	keep_insulin	< LBGT; falling	not falling; >Th11	H1	

$*HBGT = 95mg/dL, LBGT = 160mg/dL$ [24] [25]

B. Closed-Loop System Simulation

As shown in Table I, the logic rules extracted from the STPA context table are based on undefined thresholds (e.g., Th1 in row 1) that should be refined based on the target controller and adjusted using data from the patient. In other words, the inclusion of system context is driven not only by the STPA model but also by the data collected from systems.

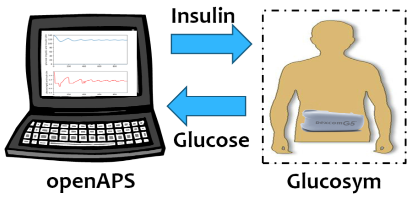


Fig. 6. Closed Loop System with OpenAPS Controller and GlucoSym.

To achieve this goal, we developed a closed-loop system by integrating the OpenAPS control software with the Glucosym patient simulator as shown in Figure 6 and collected data traces from the closed-loop system simulations using a population of patients with different characteristics and body parameters [21] (see section VII-A and VII-B for more details). These data traces were then used for parameter tuning of the logic rules extracted from the STPA context table. For each patient, we collected 13 data traces with 1500 minutes worth of run-time in the real world. Each simulation started with a different initial value for the BG level. We used data from 7 patients (13*7 data traces) for learning the threshold in the final context table (Table. I), using the Temporal Logic Extractor (TeLEX), a temporal logic synthesis tool developed by Jha et al. [26].

C. Temporal Logic Extraction

The benefit of adopting TeLEX here is that it can learn the unknown parameters such that the rules are satisfied by all the provided traces, and it is tight when only positive examples are provided. As the first step of parameter learning, we converted all the rules of the context table into the format of linear temporal logic (LTL) [27]. The set of rules described in the LTL format are shown in Table II.

To learn the thresholds, we further modified the format of the LTL rules before feeding into TeLEX. The idea is to negate the predicate that we want to learn the value of. For instance, in rule 1, we want to learn the value of IOB, less than which decreasing insulin is an unsafe action. For that, we negate the predicate (IOB<Th1) to (IOB>Th1) and learn the value of IOB greater than which the decreasing rate is allowed while satisfying the constraints: BG is greater than Target, BG rises, IOB falls and rate decreases. As a result once we learn the value of IOB in a fault-free system greater than which decrease_insulin is allowed, the monitor now can checks that with the same context, if IOB is less than the learned value, decrease_insulin is not permitted and if there is decrease_insulin command provided by the controller in the same context, the monitor will treat that as an unsafe control action and generate an alarm.

D. Safety Monitor Synthesis

After the previous three steps: STPA hazard analysis, closed-loop system simulation, and the temporal logic extraction, a set of rigorous safety requirements are achieved. These safety rules shown in table II are then synthesized in the safety monitor. The monitor continuously checks the safety requirements, any breach of which will cause the monitor to generate an alarm indicating an unsafe control action that might lead to accidents (hypo- or hyper-glycemia) in patient's body. As an example, the safety rule 10 in Table II sates that when BG is below $Th21$, then always zero_insulin command is expected. If the controller does not send a zero_insulin command while BG$<$ $Th21$ then the monitor finds the breach of the safety rule and generates an alarm. The same concept is valid for all the safety rules.

E. Preemptive Mitigation Implementation

The eventual goal of the monitor is to offer a reference to a mitigator, locating between the monitor and the actuator. When the monitor detects any unsafe action of the controller, the mitigator will stop this command, revise it, and then deliver to the actuator. For example, decrease insulin when it is more than needed, or add suitable insulin when it is insufficient. The mitigator aims to prevent as many accidents as possible and shouldn't cause any new accident at the same time.

One possible strategy to mitigate and prevent an upcoming hazard is shown in Algorithm 1.

Algorithm 1: Mitigation algorithms

```
initialization;
while not end of simulation do
    Read current BG value and output of monitor and
      controller;
    if Alerts for extra insulin input then
        $mitigation\_H1 = True$;
    else if Alerts for insufficient insulin input then
        $mitigation\_H2 = True$;
    end
    if mitigation_H1 then
        $insulin = 0$;
    else if mitigation_H2 then
        $insulin += f(IoB)$;
    end
    if $\Delta insulin < 0 or BG > BGT$ then
        mitigation_H1 = False;
    else if $\Delta insulin > 0 or BG < BGT$ then
        mitigation_H2 = False;
    end
end
```

In this paper, we only focus on the detection of unsafe control actions to enable the preemptive mitigation of hazards before they happen in the physical layer, and here we show a possible instance that the relief can be deployed. However, the effective design and evaluation of mitigation strategies that are timely and reliable is beyond the scope of our paper.

IV. Experimental Evaluation

A. Experimental Setup

We ran our experiments on a virtual machine of Ubuntu 14.04 LTS. It takes 3-5 seconds to run one iteration of the

TABLE II
TEMPORAL LOGIC RULES EXTRACTED FROMSTPA CONTEXT TABLE AND SYSTEM SIMULATION

Rule No.	**Description**
1	$\square(((BG>HBGT)\wedge(\Delta BG>0))\wedge((\Delta IOB<0)\wedge(IOB<Th1))\Rightarrow\neg\ \text{decrease\_insulin})$
2	$\square(((BG>HBGT)\wedge(\Delta BG>0))\wedge((\Delta IOB=0)\wedge(IOB<Th2))\Rightarrow\neg\ \text{decrease\_insulin})$
3	$\square(((BG>HBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB>0)\wedge(IOB<Th3))\Rightarrow\neg\ \text{decrease\_insulin})$
4	$\square(((BG>HBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB<0)\wedge(IOB<Th4))\Rightarrow\neg\ \text{decrease\_insulin})$
5	$\square(((BG>HBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB=0)\wedge(IOB<Th5))\Rightarrow\neg\ \text{decrease\_insulin})$
6	$\square(((BG<LBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB>0)\wedge(IOB>Th6))\Rightarrow\neg\ \text{increase\_insulin})$
7	$\square(((BG<LBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB<0)\wedge(IOB>Th7))\Rightarrow\neg\ \text{increase\_insulin})$
8	$\square(((BG<LBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB=0)\wedge(IOB>Th8))\Rightarrow\neg\ \text{increase\_insulin})$
9	$\square(((BG>HBGT)\wedge(IOB<Th9))\Rightarrow\neg\ \text{zero\_insulin})$
10	$\square((BG<Th21)\Rightarrow\text{zero\_insulin})$
11	$\square(((BG>HBGT)\wedge(\Delta BG>0))\wedge((\Delta IOB<=0)\wedge(IOB<Th10))\Rightarrow\ \neg\text{keep\_insulin})$
12	$\square(((BG<LBGT)\wedge(\Delta BG<0))\wedge((\Delta IOB>=0)\wedge(IOB>Th11))\Rightarrow\neg\ \text{keep\_insulin})$

simulation, which represents 5 minutes in the real OpenAPS control system. We ran 200 iterations for each test, which in total equals to 16 hours in reality. We used ten patient models with different weights and IOB susceptibilities or rates of insulin consumption as our test cases. The performance of OpenAPS is assessed in the presence of transient faults with diverse duration. And the performance of the monitor in the timely detection of unsafe commands is evaluated in both fault-free and faulty scenarios and comparison to a baseline monitor with no context awareness, and a context-aware monitor without learned thresholds. At last, we added a mitigator between the monitor and the pump, reran the simulation, and compared the effectiveness of both proposed monitor and baseline monitor.

B. Fault Injection

As a critical component of a MCPS controller, input and output data play a significant role in the decision making of the OpenAPS. Therefore, to test the robustness of OpenAPS and performance of our monitor, we injected faults with different duration and start time (two randomly chosen time from [0,100] and [100,200] iteration), into the variables storing the input data of the controller, that is the glucose feedback values from CGM, and the output commands delivered from the controller to the pump. These high-level faults were injected to represent low-level hardware faults, like bit flips in memory or disconnection between sensors, controller and actuators, that might lead to erroneous state values in the controller. According to the context table introduced in the previous section, we injected ten types of faults to the control software:

- Stop the refresh of the glucose value or insulin output
- Double/ half the glucose value or insulin output, to simulate one bit-flip at the exponent section of the relative float memory register
- Decrease/ increase the glucose value or insulin output by 2 power n (n belongs to [0,1,2,...,8] while keeping the changed value inside acceptable range)

In this paper, the fault model considers hardware errors like memory failure or poor interface connection, but it can also reflect the consequence of software attacks [28] [29]. The effectiveness and accuracy of the monitor in the detection of hazard and unsafe actions were evaluated with both fault-free and faulty data. To be specific, we trained our monitor without faults on 7 patients, and then injected faults to the 3 remaining patients, patient A, H, and J, with different initial glucose values from 80 to 200.

The total simulation numbers of faulty and fault-free cases are 4032 (3 patients x 7 initial values x 24 fault sub-types x2 start time x 4 duration) and 21 (3 patients x 7 initial values) respectively. A detailed introduction of fault scenarios and experiment numbers will be presented in Section IV-F (see Table IV).

C. Hazard Labeling

We exploited the notion of the Risk Index (RI) [22] that captures both the glucose variability and its associated risks for hypo- and hyperglycemia, to label the data points in our simulation traces as normal or hazardous and create the ground truth for verifying the safety monitor. To be specific, if the potential risk of low- or high-glucose crosses a certain threshold (5 and 9) and keeps increasing, we mark that data point as hazardous.

D. Context-Aware Monitoring

To evaluate the importance of considering the context in safety monitoring, we did a comparative analysis between our monitor and a baseline monitor. This baseline monitor was implemented based on four data authenticity rules described in the recent work [6]. These rules were defined based on the medical literature and without considering the notion of safety context, which is the main focus of our monitor synthesis technique. Another main difference is that our monitor relies on run-time simulation for prediction of the next glucose level in the patient upon execution of a given insulin command to indicate whether the command might lead to a hazard or not. Table III shows the rules used to implement the baseline monitor.

The first rule in table III indicates that BG must remain in the range of 70 mg/dL to 180 mg/dL. The second rule states that BG cannot increase by more than 3 mg/dL/min or decrease by more than 5 mg/dL/min. The third rule represents the fact that BG should not stay below the 10th percentile threshold for more than alpha minutes. Similarly, rule 4 indicates that

TABLE III
RULES OF BASELINE MONITOR

Rule No.	Description
1	$\phi 1 = \Box(BG > 70) \wedge (BG < 180))$
2	$\phi 2 = \Box((\Delta BG > -5) \wedge (\Delta BG < 3))$
3	$\phi 3 = ((BG < \lambda_{10}) \Rightarrow \Diamond_{[0,\alpha]}(BG > \lambda_{10}))$
4	$\phi 4 = ((BG > \lambda_{90}) \Rightarrow \Diamond_{[0,\alpha]}(BG < \lambda_{90}))$

BG should not remain above the 90th percentile threshold for more than alpha minutes.

E. Evaluation Metrics

- ***Hazard Coverage:*** The conditional probability that given an injection of a fault in the system, it leads to an unsafe state or hazardous condition.
- ***MTTH:*** Mean Time to Hazard represents the average time between activation of a fault in the system (T1 in Figure 10) until the occurrence of a safety hazard (T3). This is an important metric indicating the amount of time budget we have for the detection and mitigation of faults before they lead to safety hazards and incidents.
- ***Detection Coverage:*** Detection coverage represents the percentage of potential unsafe control actions that are detected by our monitor. We measure the detection coverage using the number of true positives (TP), true negatives (TN), FP, and false negatives (FN) as well as average micro F1 score. For instance, when a hazard happens at T3 and ends at T4, after a fault injected at T1 (Figure 7(a)), it is a TP, FP or FN if alert happens between [T1, T4] (Figure 7(b)), outside [T1, T4] (Figure 7(c)), or not happens at the whole simulation (Figure 7(d)). On the other hand, when no hazard happens in the whole simulation if an alert is generated, it is also a FP, otherwise, it is a TN.

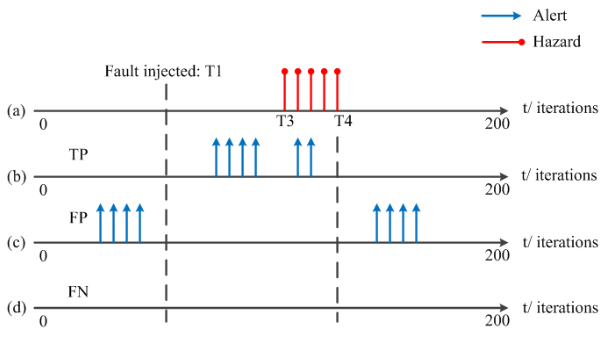


Fig. 7. Definition of TP, FP and FN when hazard happens in the experiment

- ***Detection Latency:*** We measure the time between the activation of fault until the detection of unsafe control action by the monitor as the detection latency. This metric provides us a measure of the fault propagation time in the system and also impacts the reaction or recovery time.
- ***Reaction Time:*** The time difference between a monitor alert and the occurrence of a hazard, which is the maximum time that the patient and doctors have to respond to a monitor alert and take action, before a hazard and potential harm to the patient happens.

F. Experimental Results

In this section, we present the experimental results of our case study on the OpenAPS controller. We injected different faults in the OpenAPS controller and show the efficacy of our proposed technique in terms of hazard analysis, safety monitor performance, time analysis, and preemptive mitigation.

Hazard Analysis: In Table IV we present the results of assessing the resilience of the OpenAPS controller to different fault scenarios and types. As we can see in the table, both the fault scenarios of "rate" and "glucose value" have a relatively high hazard coverage (19.1% and 11.9% respectively). This is because the "rate" directly affects the dose of insulin injected into the patients, and the glucose feedback value has a crucial influence on the decision of the OpenAPS controller, whose output is exactly the state variable "rate." On the other hand, the hazard coverage of faults injected into the state variable "rate" was higher than that of "glucose value."

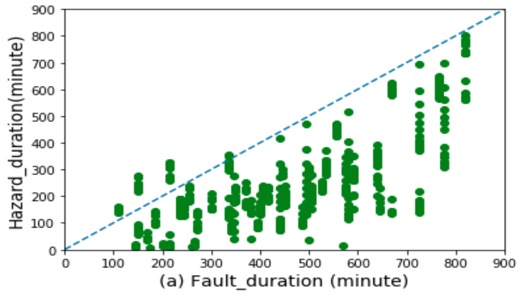
TABLE IV
HAZARD COVERAGE FOR DIFFERENT FAULT SCENARIOS

Scenarios	Fault Type	Total No. Simulations	Hazard Coverage
1	hold-rate[a]	168	0.0%
2	double-rate	336	0.0%
3	half-rate	336	4.2%
4	bitflip_add-rate	504	46.2%
5	bitflip_dec-rate	504	21.0%
Summary of Faulty Rate Scenarios		1848	19.1%
6	hold-glucose	168	0.0%
7	double-glucose	168	25.0%
8	half-glucose	168	0.6%
9	bitflip_addglucose	840	25.8%
10	bitflip_decglucose	840	0.0%
Summary of Faulty Glucose Scenarios		2184	11.9%

[a]Stop the refresh of insulin rate due to connection or memory error.

We also observed that scenario "hold-rate" and "hold-glucose" didn't cause any hazard, meaning lost connection with the sensors for less than 11 hours when the glucose is in the normal range is not a severe situation from our experiment. The same situation happens to the scenario "double-rate" and "bitflip_decglucose." For "double-rate," it is because when BG is low, the initial rate is also pretty small, so is the doubled rate, which can't change the BG significantly. As for the "bitflip_decglucose," the reason is that the rate value and glucose level have already back to normal when the fault is injected.

To further evaluate the accuracy of RI in predicting hypo- and hyper-glycemia events and validate the reasonableness of fault model, we analyzed all the hazardous cases and found for 97.6% the BG value is not in the safe range of [70,280].This implies that not all hazards or unsafe systems (indicated by risk index) might lead to accidents (hyper- or hypo-glycemia), but RI has a high accuracy in representing accidents. Besides, we compared the duration and time relation between hazards and faults in Figure 8. Figure 8(a) shows that in most hazardous cases, the hazard duration is shorter than the fault duration, and it increases with an increase in duration of fault. Figure 8(b) shows that in most cases, the hazards will last for a while after



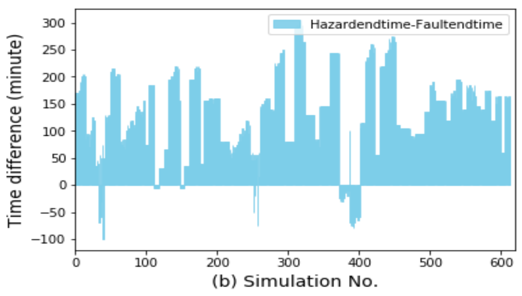


Fig. 8. Duration and Ending Time of Faults and Hazards

the end of fault period, as it takes some time for the controller to bring the BG value back to normal. These results imply the fact that it takes a while for the faults to get propagated into the system and cause hazards, and the impact of faults on the system state will last even after removing the fault from the system.

Safety Monitor Performance: We present the accuracy of context-aware with learned threshold (referred as context-aware or context-aware (WT) monitor), context-aware but without learned threshold (referred as WOT monitor), and baseline monitor using Receiver Operating Characteristic (ROC) curve in Figure 9, where the red dot line shows the performance of our proposed monitor with the low BG target (LBGT) value and high BG target (HBGT) value in Table I ranging from 80 to 110 [24] and from 140 to 180 respectively [25]. As we can see, (1) for most of the target values, Context-aware monitor has a better performance than the baseline (blue dot), and WOT monitor (yellow line); (2) in some cases (HBGT=145mg/dL) baseline and WOT have lower FPR than Context-aware monitor. But when taking into account that the OpenAPS keeps a stable BG around [145,150], these cases should not be considered as a reasonable choice of HBGT in our simulation. As an example, we choose an average BGT (LBGT=95, HBGT=160) as the threshold of our context-aware monitor (green diamond), whose detailed performance is shown in Table V.

We can see that, for the same number of simulations (4032) and hazards, the proposed context-aware monitor generates a much fewer number (475) of alarms than the baseline monitor, which leads to more than a half lower false alarm rates while keeping the F1 score high (more than 0.89).

For the fault-free experiment, it is also the case that overall the context-aware monitor performs better than the baseline. We can see that the baseline monitor generates 3 alarms while

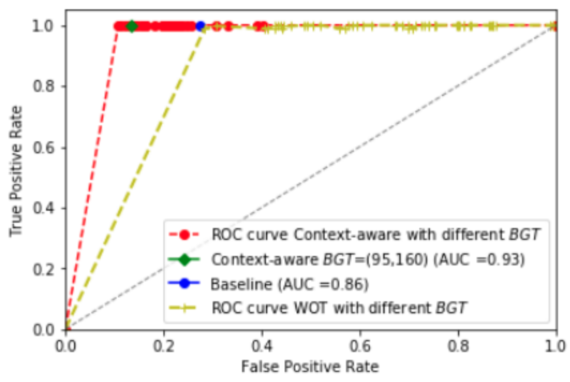


Fig. 9. Receiver Operating Characteristic Curve

there is no hazard. This causes to have a high percentage of false alarms (14.3%) when using the baseline monitor. On the other hand, the context-aware monitor generates no false alarm and ensures an overall higher F1 score.

When looking into all the alert message generated by the baseline, rule 3 and rule 4 doesn't take effect at all, which is probably because the OpenAPS controller does not let the glucose value to exceed the thresholds for a long period as indicated by rules 3 and 4. This further shows the drawback of the baseline monitor in not taking into account the controller actions.

The purpose of presenting the WOT monitor here is to emphasize the importance of the learning process. Without learning the threshold of glucose and insulin rate, the WOT monitor exhibited the lowest TP and highest FP among these monitors, though owning the awareness of the context.

To sum up, the context-aware monitor outperforms the baseline monitoring with a 15.6% increase in average F1 score, and 50.6% reduction in the number of FP alarms (11.8% of total simulation numbers) for faulty cases, and with a 16.3% increase in average F1 score for fault-free cases.

Time Analysis: In Figure 10, we present the fault propagation time for baseline, WOT and context-aware (WT) monitors. We have three main observations here:

(1) The context table helps to set up a more severe situation to test monitors.

(2) The learning process with Telex promotes more accurate detection of unsafe control action and decent of unnecessary disturbing alerts.

(3) The context-aware monitor has the best overall performance.

Before we start the discussion, we want to emphasize that the human body is such a large lag system that it usually takes hours to digest food and for the insulin to bring BG value back to normal from a severe situation. Therefore keeping hours of reaction time (instead of seconds or minutes in other CPS) matters in APS, and the longer, the better. We also traversed all the fault-free data traces with an initial BG value outside [120,140] mg/dL and found it takes about 167.1 minutes on average for the openAPS to bring BG value back around target value.

TABLE V
SUMMARY RESULTS FOR MONITOR PERFORMANCE IN FAULTY AND FAULT-FREE SCENARIOS

Fault Scenarios	**Faulty**			**Fault-free**		
No. Simulations	4032			21		
No. Hazards (%)	613 (15.2%)			0		
Monitors	**Baseline**	**WOT**	**Context-Aware (WT)**	**Baseline**	**WOT**	**Context-Aware (WT)**
No. Alerts (%)	1551 (38.5%)	3007 (74.6%)	1076 (26.7%)	3 (14.3%)	13 (61.9%)	0
True Negatives (%)	2481 (61.5%)	1025 (25.4%)	2956 (73.3%)	18 (85.7%)	8 (38.1%)	21 (100.0%)
True Positives (%)	613 (15.2%)	613 (15.2%)	613 (15.2%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
False Positives (%)	938 (23.3%)	2394 (59.4%)	463 (11.5%)	3 (14.3%)	13 (61.9%)	0 (0.0%)
False Negatives (%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Micro F1 Score	0.77	0.41	0.89	0.86	0.38	1
Recovery Rate	49.8%		82.1%			

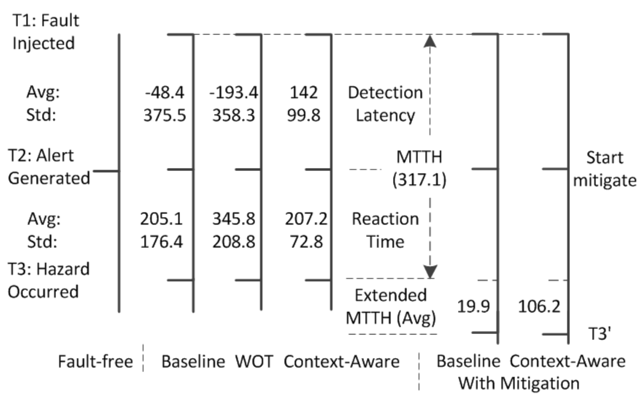


Fig. 10. Average Fault Propagation Time for Different Scenarios (Unit: minutes)

From Figure 10, we can observe the performance of each monitor in faulty cases, while no hazard happens in fault-free cases, which means fault injection technique helps to imitate severe situation to examine the monitors' performance. However, even in such an adverse scenario, the context-aware monitor can still timely detect unsafe control commands and leave enough time (nearly 3.5 hours, 2.1 minutes longer than baseline) for patients and physicians to take any action in response. Moreover, the context-aware monitor owns a much less (58.7% and 73.4%) standard deviation in reaction time and detection latency than the baseline monitor, representing a more stable performance on ensuring safe reaction time for the patients.

We also observe that baseline monitor has a large negative average detection latency with a high standard deviation, which means it either generates alerts too early (before a fault is injected), that is to say, it generates much more unnecessary alerts also underlined by the high FP in Table V, or too late to have enough reaction time. When we look at the WOT monitor in Figure 10, it has a much longer reaction time than baseline, but keeps a much lower negative detection latency, reflecting the benefit of aware context and also exhibiting the drawback of the unknown threshold.

To conclude, with the learned threshold, the context-aware monitor outperforms the other two monitors in FP and the accuracy on predicting unsafe control actions while keeps a safe reaction time with a much less standard deviation.

Preemptive Mitigation: The experiment result shows that only 110 hazards occurred after adding the mitigation function into the 613 TP cases in Table V, keeping a recovery rate of 82.1%, and that no new hazard was introduced into the system after adding the mitigation function to the 463 FP cases. Though there are still 110 TP we didn't prevent, we extended the reaction time for 106.2 minutes on average. In comparison, the recovery rate of the Baseline monitor is 49.8% and it extended the average hazard time by 19.9 minutes, while in 4 cases hazard happened 5 minutes earlier than that without mitigation, which again exhibits the disadvantage of not having stable performance in keeping safe reaction time and of unawareness of the context of the patient.

V. RELATED WORK

Run-time Verification: Run-time verification and assurance of safety properties based on formal descriptions and models of systems have been an active area of research in safety-critical systems [7] [8] [9] . The safety properties are usually described as Finite State Machines (FSM) or LTL and synthesized into run-time monitors [10] [11]. The most relevant and recent work in this area is [12], which presents robust online monitoring of signal temporal logic (STL) in CPS. Our work distinguishes from previous work in incorporating the notion of safety context in MCPS and the information from the hazard analysis process for the specification of safety properties and synthesis of run-time monitors.

Safety and Security Monitoring in CPS: Another relevant area of research is anomaly detection in CPS to mitigate security attacks. Alemzadeh et al. [4] presented dynamic model-based detection and mitigation of safety-critical attacks on CPS in robotic surgery. Aliabadi et al. [30] proposed a dynamic system property mining technique to infer likely specifications based on observational data. They used the technique to mine temporal properties from the smart meter and medical devices to enable intrusion detection.

Others have studied techniques for safety monitoring and risk mitigation in CPS. For example, [31] presented a probabilistic detection method for risk management and mitigation in smart MCPS. The safety monitoring approach studied in [32] identifies the safety functions and separates them from the main application ones. [33] applied real-time simulation for run-time verification of CPS. [34] introduced the notion of safety guards for enforcing safety properties in CPS. [35] presented an adaptive fault tolerance framework for CPS. [36] illustrated a monitoring framework for autonomous systems with an active strategy to maintain safety.

Instead of using model-based detection which doesn't work well in APS, as the human body is a pretty slow and large lag system with unpredictable variance like eating food or taking exercises, we use STPA hazard analysis to identify safety properties. Besides, by just knowing the spec and safety requirements from STPA, the monitor can be synthesized and doesn't need to have access to the internals of the device or code. Thus, the monitor can be seamlessly integrated as a wrapper and is generic to any MCPS.

Fault Detection and Tolerance in Medical Devices: One of the most related areas to this work is the model-based development and resilience by the construction of MCPS. In [37], a generic formal model of infusion pump systems was proposed. The authors first conducted a rigorous requirement and hazard analysis for infusion pumps and then offered a generic model that ensures all the safety requirements to avoid the possible hazards. In [38], a "Correct by construction" approach was presented for the trustworthy development of a hemodialysis machine. In [39], the authors proposed a formal approach for early validation of the MCPS. A fuzzy logic-based fault detection technique for wearable medical devices was introduced in [40]. Young et al. [6] stated a data authenticity monitoring system to ensure the authenticity of the collected data in a body area network assembled to provide treatment to the Type 1 diabetic patient. S. Pinisetty et al. [41] introduce the concept of synchronous run-time enforcement of reactive systems. However, these approaches either are unaware of the context of the patient or don't consider the reaction of the controller. Besides, we use the fault inject technique to decrease the experiment scale and accelerate the validation process of the proposed monitor.

VI. Conclusion

In this paper, we proposed an approach for automated synthesis of context-aware safety monitors for MCPS from the high-level requirements identified during the hazard analysis process. To the best of our knowledge, our hybrid model- and data-driven synthesis approach is the first attempt at using the information from the STPA hazard analysis process to generate safety checking rules that can detect and prevent unsafe control commands in MCPS control software. Our experiments using a case study of a closed-loop artificial pancreas system in both fault-free and faulty scenarios (transient and permanent faults affecting the OpenAPS control software) show preliminary encouraging results on the ability of monitor in accurate and timely detection of unsafe control actions and hazards.

An excellent future direction might be an online learning approach that will continuously learn the threshold from the patient who uses the monitor. Future work might also include focusing on defining more suitable context-checking intervals, learning patient-specific thresholds using a larger population of patients, and evaluating the applicability of the proposed synthesis approach on a broader range of MCPS, in particular, those with tighter timing and resource constraints.

Code Availability

The code for the closed-loop APS system and the safety monitor is available at the following online repository: [Removed for anonymous submission]

References

[1] H. Alemzadeh, R. K. Iyer *et al.*, "Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis," *IEEE Security & Privacy*, vol. 11, no. 4, pp. 14–26, 2013.

[2] D. Halperin, T. S. Heydt-Benjamin *et al.*, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 129–142.

[3] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*. IEEE, 2011, pp. 150–156.

[4] H. Alemzadeh, D. Chen *et al.*, "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 395–406.

[5] H. Alemzadeh, C. Di Martino *et al.*, "Towards resiliency in embedded medical monitoring devices," in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*. IEEE, 2012, pp. 1–6.

[6] W. Young, J. Corbett *et al.*, "Damon: A data authenticity monitoring system for diabetes management," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2018, pp. 25–36.

[7] I. Lee, S. Kannan *et al.*, "Runtime assurance based on formal specifications," *Departmental Papers (CIS)*, p. 294, 1999.

[8] G. Eakman, C. Pit-Claudel *et al.*, "Correct-by-construction implementation of runtime monitors using stepwise refinement," in *Dependable Software Engineering. Theories, Tools, and Applications: 4th International Symposium, SETTA 2018, Beijing, China, September 4-6, 2018, Proceedings*, vol. 10998. Springer, 2018, p. 31.

[9] Falcone, YliÃÍs *et al.*, "What can you verify and enforce at runtime?" *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 3, pp. 349–382, 2012.

[10] H. Barringer, A. Goldberg *et al.*, "Rule-based runtime verification," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*. Springer, 2004, pp. 44–57.

[11] K. Havelund and G. Roşu, "Synthesizing monitors for safety properties," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2002, pp. 342–356.

[12] J. V. Deshmukh, A. Donzé *et al.*, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, 2017.

[13] N. Leveson and J. Thomas, "An stpa primer," *Cambridge, MA*, 2013.

[14] P. Asare, J. Lach, and J. A. Stankovic, "Fstpa-i: A formal approach to hazard identification via system theoretic process analysis," in *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, April 2013, pp. 150–159.

[15] J. Thomas, "Analysis of safety-critical computer failures in medical devices," *Technical Report SAND2012-4080*, 2012.

[16] G. Schirner, D. Erdogmus *et al.*, "The future of human-in-the-loop cyber-physical systems," *Computer*, vol. 46, no. 1, pp. 36–45, 2013.

[17] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011.

[18] U. Food and D. Administration, "Maude adverse event report: Philips medical systems philips intellivue x2 portable patient monitor, mdr report 2154693," https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=2154693, 2010.

[19] H. Alemzadeh, “Data-driven resiliency assessment of medical cyber-physical systems,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2016.
[20] “Openaps reference design,” https://openaps.org/reference-design.
[21] “Glucosym,” https://github.com/Perceptus/GlucoSym.
[22] W. Clarke and B. Kovatchev, “Statistical tools to analyze continuous glucose monitor data,” *Diabetes technology & therapeutics*, vol. 11, no. S1, pp. S–45, 2009.
[23] B. P. Kovatchev, “Metrics for glycaemic controlâĂŤfrom hba 1c to continuous glucose monitoring,” *Nature Reviews Endocrinology*, vol. 13, no. 7, p. 425, 2017.
[24] A. M. Hersh, E. L. Hirshberg *et al.*, “Lower glucose target is associated with improved 30-day mortality in cardiac and cardiothoracic patients.” *Chest*, vol. 154, no. 5, pp. 1044–1051, 2018.
[25] A. Association of Clinical Endocrinologists and A. Diabetes Association, “Standards of care in diabetes,” *Diabetes Care*, vol. 33, no. 1, pp. 11–61, 2010.
[26] “Temporal logic extractor,” https://github.com/susmitjha/TeLEX.
[27] “Linear temporal logic,” https://en.wikipedia.org/wiki/Linear_temporal_logic.
[28] H. Lin, H. Alemzadeh *et al.*, “Challenges and opportunities in detection of safety-critical cyber-physical attacks,” in *IEEE Computer Magazine*. IEEE, 2019.
[29] ——, “Safety-critical cyber-physical attacks: Analysis, detection, and mitigation,” *Symposium and Bootcamp on the Science of Security (HOTSOS)*, 2016.
[30] M. R. Aliabadi, A. A. Kamath *et al.*, “Artinali: dynamic invariant detection for cyber-physical system security,” in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017, pp. 349–361.
[31] A. Rao, N. Carreon *et al.*, “Probabilistic threat detection for risk management in cyber-physical medical systems,” *IEEE Software*, vol. 35, no. 1, pp. 38–43, 2017.
[32] L. Masson, J. Guiochet *et al.*, “Synthesis of safety rules for active monitoring: Application to an airport light measurement robot,” 04 2017, pp. 263–270.
[33] X. Zheng, C. Julien *et al.*, “Real-time simulation support for runtime verification of cyber-physical systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 4, p. 106, 2017.
[34] M. Wu, H. Zeng *et al.*, “Safety guard: Runtime enforcement for safety-critical cyber-physical systems,” in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.
[35] Y. Xu, I. Koren, and C. M. Krishna, “Adaft: A framework for adaptive fault tolerance for cyber-physical systems,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 3, p. 79, 2017.
[36] M. Machin, J. Guiochet *et al.*, “SMOF - a safety MOnitoring framework for autonomous systems,” vol. 48, no. 5, pp. 702–715. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01394139
[37] D. E. Arney, R. Jetley *et al.*, “Generic infusion pump hazard analysis and safety requirements version 1.0,” *Technical Reports (CIS)*, p. 893, 2009.
[38] A. Mashkoor and M. Biro, “Towards the trustworthy development of active medical devices: a hemodialysis case study,” *IEEE Embedded Systems Letters*, vol. 8, no. 1, pp. 14–17, 2015.
[39] L. Silva, H. Almeida *et al.*, “A model-based approach to support validation of medical cyber-physical systems,” *Sensors*, vol. 15, no. 11, pp. 27 625–27 670, 2015.
[40] C. C. Oliveira and J. M. da Silva, “A fuzzy logic approach for highly dependable medical wearable systems,” in *2015 IEEE 20th International Mixed-Signals Testing Workshop (IMSTW)*. IEEE, 2015, pp. 1–5.
[41] S. Pinisetty, R. Partha S *et al.*, “Runtime enforcement of cyber-physical systems,” *International Conference on Embedded Software (EMSOFT) 2017*, vol. 16, no. 5, pp. 178:1–178:25, 2017.

VII. Appendix

A. OpenAPS

The OpenAPS is a safe and advanced APS controller that is developed by an open-source community [20]. It adjusts the insulin delivery of an infusion pump to automatically keep the BG level of the diabetic patient within a safe range. The internal architecture and necessary input-output connections of OpenAPS are shown in Figure 11. The shaded region indicates the OpenAPS controller. And the "File storage" section reflects the behavior of the insulin pump. The Determine_basal process accepts profile, IOB, BG, and current insulin delivery (temp_basal) and calculates the suggested insulin delivery to the patient.

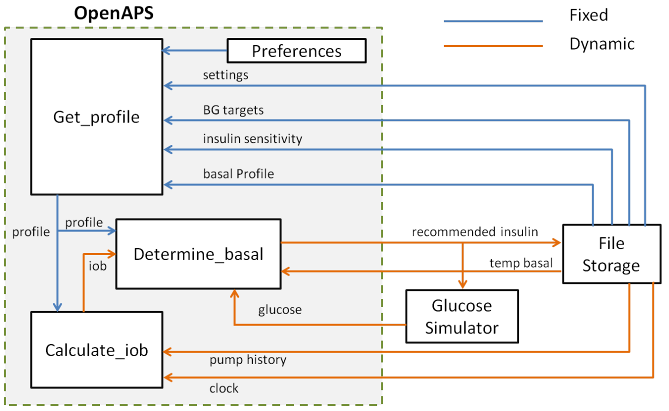


Fig. 11. OpenAPS Architecture and Input Output.

More specifically, OpenAPS collects the previously delivered insulin amount, combined with the duration of the activity, and it calculates the net IOB. Using the glucose sensor readings, OpenAPS then calculates the eventual BG using the following equation:

$$CurrentBG - ISF * IOB = eventualBG \qquad (1)$$

where $CurrentBG$ is the current BG, ISF is the Insulin Sensitivity Factor,and $EventualBG$ is the estimated BG by the end of current insulin delivery.

While current BG is below a threshold value, OpenAPS continues to issue temporary insulin delivery to zero until BG is not rising. Otherwise, OpenAPS determines whether the glucose values are rising or falling more than expected. In that case, it performs the course of actions shown in Algorithm 2.

Algorithm 2: OpenAPS algorithms

```
if BG is rising, but $E\_BG < BG\_Target$ then
    cancel any temp basal;
else if BG is falling, but $E\_BG > BG\_Target$ then
    cancel any temp basal;
else if $E\_BG > BG\_Target$ then
    cancel 30min temp basal;
    if recommended temp>existing basal then
        issue the new high temp basal;
    else if recommended temp<existing basal then
        issue the new high temp basal;
    else if 0 temp for >30m is required then
        extend zero temp by 30 min;
    end
end
```

TABLE VI
PATIENT PROFILES

Patient	BW	C_I	τ_1	τ_2	V_G	$1/p_2$	EGP	I_{EFF}
A	89	20.1	49	47	253	0.01	1.33	0.002
B	63	12.81	41	10	261	0.01	0.6	0.004
C	65	9.09	71	70	199	0.02	1.07	0.003
D	116	18.13	91	70	337	0.008	0.98	0.00002
E	64	15.35	46	46	188	0.01	0.6	0.004
F	51	5.88	68	30	104	0.009	0.603	0.001
G	77	18.06	60	60	263	0.01	1.11	0.0023
H	65	5.4	95	37	137	0.01	1.3	$1e^{-8}$
I	100	8.75	131	21	193	0.01	1.27	0.006
J	64	13.09	53	53	204	0.01	0.6	0.001

BW=Body Weight.
C_I=insulin clearance (dl/min).
τ_1, τ_2=Time constant associated with insulin movement between the SC delivery site and plasma (min).
V_G=Distribution volume in which glucose equilibrates (dl).
$1/p_2$=Delay in insulin action upon increase in plasma insulin (1/min).
EGP=Endogenous glucose production rate that would be estimated at zero insulin (mg/dl/min).
I_{EFF}=Effect of glucose per se to increase glucose uptake into cells and lower endogenous glucose production at zero insulin (1/min).

B. GlucoSym Patient Simulator

GlucoSym is an open-source human body glucose simulator that is developed to help build and test automatic insulin delivery systems. Table VI shows the patient population used in our simulations along with their corresponding body parameters.