CS 6501 Project Report – Fuzzing for CPS: *Fuzzing a Heart Model*

Team Fumblebuzz

ALAN WANG, ARON HARDER, INGY ELSAYED-ALY, and JOSEPHINE LAMP

1 ABSTRACT

In this paper we explore two different types of heart models within Matlab Simulink as a way to extend the concept of fuzzing into the realm of Cyber-physical systems. Specifically, we explored 1) a pacemaker model which paces the atrioventricular node and its relationship via conduction with the sinoatrial node, and 2) a Heart Systemic Pulmonary (HSP) model that models the human cardiovascular system, including the pulmonary and systemic circulatory systems. Our experiments successfully generated inputs that resulted in invalid model behavior for both models, validating the effectiveness of input range scaling techniques to identify parameter boundaries for valid behaviors.

2 INTRODUCTION

2.1 Motivation

Cyber-physical system (CPS) devices are becoming more and more common in our everyday lives. Many of these devices perform safety-critical functions, including medical devices such as pacemakers and self-driving cars. These devices are often designed using formal methods in such a way as to verifiably guarantee the safety of the operator. Despite this "safe by construction" approach, bugs still present themselves in these devices with catastrophic results. Our project aims to put this "safe by construction" assumption to the test by searching for bugs in verifiable models. In this case, we use fuzzing techniques to attempt to uncover bugs for two different CPS models.

2.2 Background

Fuzzing typically involves running random text or file inputs through a program, and analyzing program crashes for exploitability. Such methods do not work when fuzzing CPS devices. CPS devices do not usually use text or files as inputs, but an array of sensors that output many different types of values representing a variety of measurements. These sensors have some range of valid values, meaning that fuzzed inputs need to be restricted to these ranges. Additionally, although a CPS device can crash, it may violate safety regulations and its initial design specification in other ways. For example, a pacemaker may induce an endless pacing loop where the heart cannot recover its own beat if certain conditions are met. These challenges make it necessary to have a strong understanding of the device being fuzzed in order to define both valid input parameters and unsafe outputs. For safety and practicality reasons, we decided to use the Simulink modeling and development tools for systems.

2.2.1 CPS Fuzzers. We initially performed a survey of existing specialized CPS fuzzers. The first CPS fuzzers we considered are CyFuzz and its derivative SLforge [2, 3]. Unfortunately, both fuzzers target the Simulink development tools rather than the CPS model itself. They generate Simulink block model files to perform fuzzing of the CPS development environment. Then we considered another CPS fuzzing tool called DeepFuzzer [1]. Deep Fuzzer uses deep learning to generate CPS models, once again aimed at finding bugs in CPS development environments rather than in the model behavior. Unfortunately, all of the existing CPS fuzzers attempt to fuzz the development environment rather than the CPS devices themselves. In our opinion, this is because it is challenging to create a general fuzzer for CPS models, the model and its application must be well understood and the fuzzing needs to be tailored to it.

2.2.2 Medical Background. In order to better understand how to fuzz heart models we first needed to understand the details of how the heart functions. The heart is generally composed of two ventricles (left and right) and two atrial nodes [5]. When the heart is paced (naturally), the electric pulse starts at the Sinoatrial node (SA) then it is felt in the Atrioventricular node (AV) after a conduction delay which is a result of the physical properties of the heart tissue between the two atrial nodes. Every heart pulse signal can be

decomposed into several time periods: the Effective Refractory Period (ERP), the Relative Refractory Period (RRP) and the Rest state. Moreover, the most important property of the heart for this project is automaticity which refers to the heart's ability to trigger itself over time.

In Fig. 1, the Electrocardiogram (ECG) as well as the membrane potential are represented for the cardiac cycle. The Membrane potential represents how much electric potential can be measured in the membrane over time. During the ERP period, the heart will not respond to an electric pulse. During the RRP period, the heart can respond to a pulse but will need a stronger electric pulse than normal to trigger the same response as during the rest period (between two heart beats).

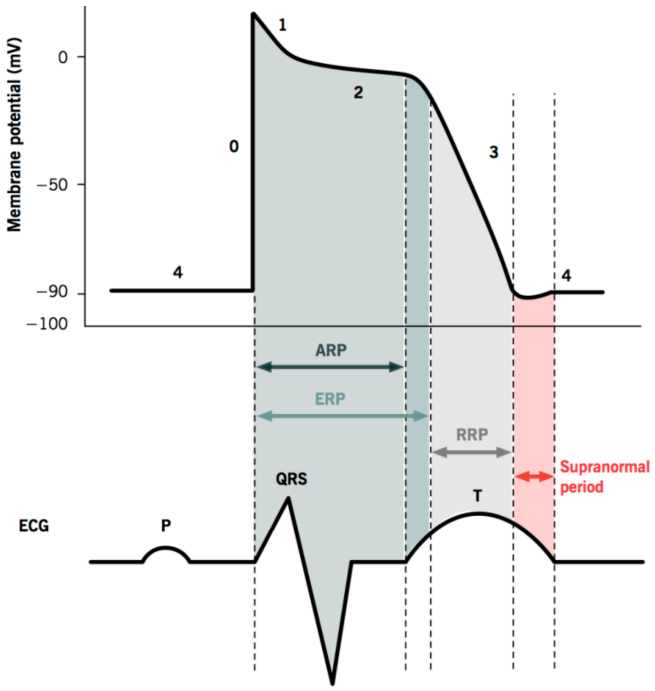


Fig. 1. Heart Refractory Period in Relation to the ECG

3 PROJECT DESCRIPTION

The goal of this project is to propose a method to use fuzzing concepts to CPS devices. CPS devices are usually modeled, verified then tested through before being manufactured. For this reason, we chose to use the popular systems tool Simulink. Our approach consists in

applying fuzzing concepts to Simulink models which serve as case studies. The first model consists of a model of the electric properties of the heart and a pacemaker. The second model simulates the pulmonary system function and the systemic function in relation to the heart function.In other to make the fuzzing realistic, we first have to understand the systems we are attempting to fuzz and what were the objectives of the model. For example, for the first model the most important function of the model is the heart rate which must remain in certain ranges. We first define the medically valid ranges for each input to our models, and for each model we define what would be an abnormal behavior based on the outputs of the model. For the purpose of this project, we developed a tool in the form of Matlab and Python scripts to generate simulation experiments, our desired inputs for each experiments and finally to detect and report errors. The main concept behind this tool is represented in Fig 2, it shows that there are three main steps, the generating of our inputs, the simulation and the detection.

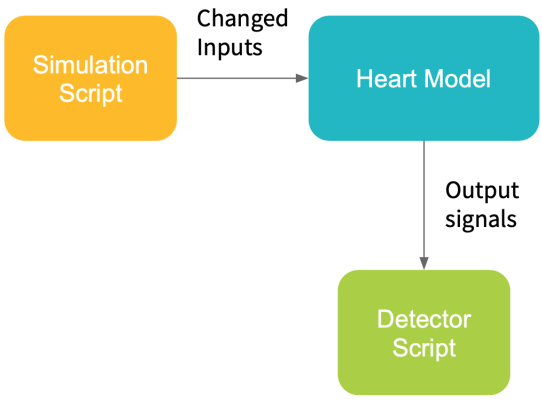


Fig. 2. Simulink Fuzzing Overview

4 FUZZING HR MODEL 1

4.1 Model Description

The first Simulink model used represents a heart and pacemaker that models the electrical properties of the heart by modeling the SA node, AV node, the conduction delay as well as an AV node pacemaker. In Fig. 3, the Simulink block representation of the model is shown. The block called "NodeLongERP1" represents the SA node, the "AtoV Path" block

represents the physical path the pulse from the SA node take to the AV node which is represented by the "NodeLongERP" block. The yellow block on the right represents the pacemaker. Note that the output "Pace" of that block which represents the pulse generated by the pacemaker is sent as an input to the AV node (thus making it an AV pacemaker). The input of the AV node is a logical OR between the pacemaker pulse and the pulse from the SA node. In the rest of this report, we refer to the pacing signal as Vpace.
This Simulink model was generated using Formal Methods techniques for modeling and verification. It was designed as a timed automata in order to take into account the different time constraints for the pacing and to detect medical issues relating to the timing of heart pulses (tachychardia, bradychardia, etc). It was then verified for correctness using Linear Temporal Logic safety specifications. Thus, errors in this model either show that the safety specification used to verify this model is insufficient or that the model was not represented correctly in the Simulink model.

Inputs. Our model uses five inputs described in table 2. The Effective Refractory Period (ERP) is the time it takes the heart to recover from an action potential. Following the action potential, the heart attempts to return to equilibrium voltage. The Relative Refractory Period (RRP) is the time it takes between when the heart starts returning to equilibrium until it reaches equilibrium. Tcond indicates the conduction time, or the time it takes for an electrical signal to travel between the heart nodes. Rest indicates the time that the heart takes to rest between heartbeats. The Lower Rate Interval (LRI) is the minimum time allowed between two VA signals by the pacemaker.

Outputs. Our model produces three outputs, SA node signal, AV node signal, and the Vpace signal. The Sinoatrial (SA) output indicates that the Sinoatrial node has produced a beat. Similarly, the Atrioventricular (AV) output indicates that the Atrioventricular node has produced a beat. Vpace indicates that the pacemaker has produced a beat.These signals allow us to calculate the heart rate in beats per minute (bpm), the number of consecutive pacemaker pulse as well as the delay from the SA to VA node.

Parameters	Minimum normal value	Maximum normal value
Rest	280 ms	340 ms
ERP	250 ms	460 ms
RRP	50 ms	100 ms
Tcond	50 ms	200 ms
LRI	1100 ms	1300 ms

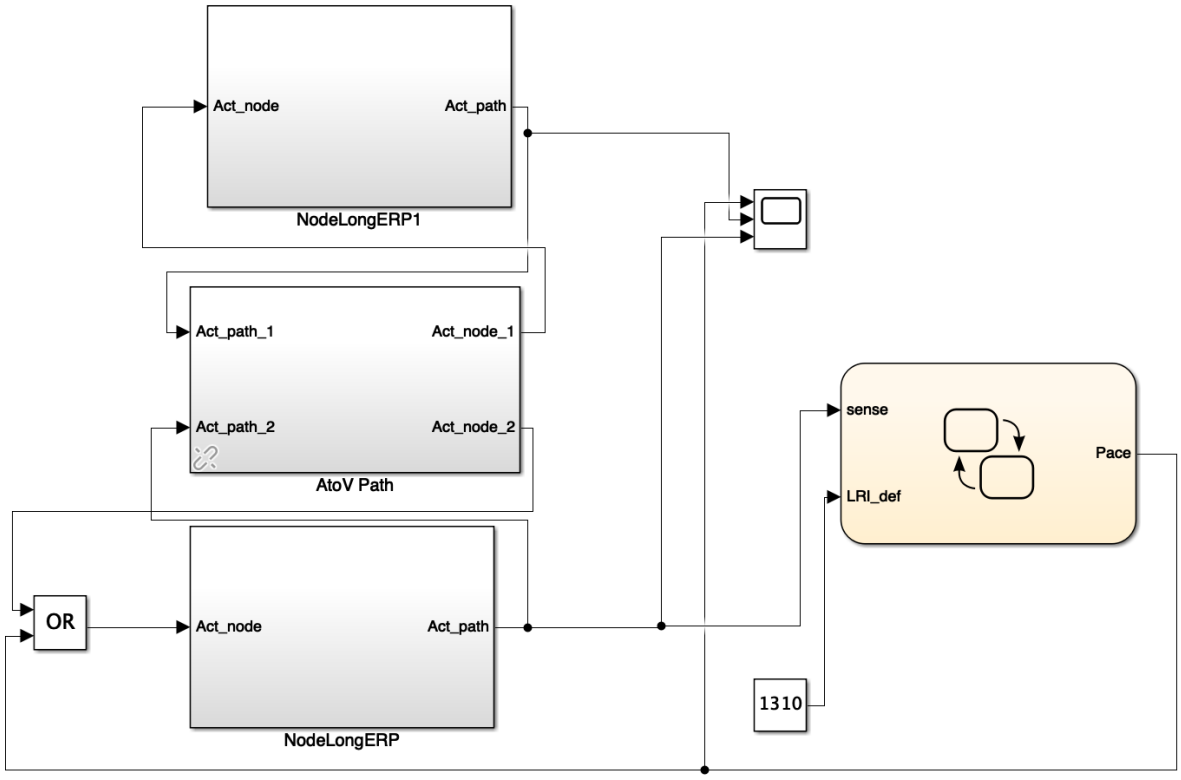


Fig. 3. Simulink Block Model

Algorithm 1 Pacemaker-Fuzzer(nExperiments)

```
****** Set Fuzzing Params ******
****** Generate Random Fuzzer Inputs ******
****** Start Simulation ******
out = get 3 output vars from model
for sliding_window in data do
    bpmVA = count(VA)/window_length
    bpmSA = count(SA)/window_length
end for
for bpm in bpmVA do
    if bpm < 60 or bpm > 90 then
        Notify user of bug found
    end if
end for
for bpm in bpmSA do
    if bpm < 60 or bpm > 90 then
        Notify user of bug found
    end if
end for
a2v = times between SA and VA
for time in a2v do
    if time > 200 || time < 50 then
        Notify user of bug found
    end if
end for
hijack = count(Vpace repetitions)
for cnt in hijack do
    if cnt > 10 then
        Notify user of bug found
    end if
end for
```

4.2 Fuzzing Overview

In order to fuzz this model, we initially started by giving random number for the different inputs of the model. However, we realized that this method is not realistic. For this reason, we searched through medical literature to understand the ratios of the inputs to each other and the normal ranges for an average person at a resting state. For some of the issues we were attempting to detect normal ranges did not produce any error. However, people who typically use pacemaker do not have normal ranges for these inputs, when they are too high then the person has bradycardia and when the time delays are too small then the person has tachycardia. Thus, in some experiments we increase the range of the numbers generated for the inputs by a factor of 20% then 100%.

In algorithm 1, we outline the pseudo-code of the tool we developed. In order to have a more accurate evolution of some of the delays calculated we use a sliding window technique. This enables to have a more detailed evaluation of the time delays over the simulation time which is (10s). From lines 1 to 4, is the part referred to as the simulation script in Fig. 2. We use it to generate random parameters for the number of experiments (nExperiments) then we start the simulation and retrieve the output signals. From lines 5 to 30, is the part of the script we call the detector script in Fig. 2. That part calculates the heart rate, the AtoV delay and the number of consecutive pacing pulses.

4.3 Fuzzing Medical Properties

Description. In order to output significant errors for this model, we need to consider medical properties of the heart and what values would make the heart to be functioning abnormally. Because this model includes a pacemaker, its primary function is to regulate the heart rate of the patient. Thus, we check for Tachycardia and Bradycardia during our experiments. We also check for endless loop pacing that would cause irritation and degradation of the heart tissue which is called rhythm hijacking. Finally, we look at the delay between when a pulse is generated in the SA node and when it is felt in the AV node because it can also give a measure of the health of the heart tissue.

Tachycardia and Bradycardia. As previously mentioned, we use a sliding window to obtain as many values as possible during the simulation time to make sure at no point during the simulation the heart rate was too high. In algorithm 1, lines 5 to 18 are used to

detect anomalous heart rates. It is important to note that because the pulse of the heart must be felt in the AV node we calculate both the heart rate felt in the SA node and the heart rate perceived at the AV node. Recall that the normal ranges for a normal person in a resting state are 60 to 90 bpm (Table 2). In Fig. 4, we show a run where no abnormal behavior is detected (left). The red lines indicate the upper and lower bounds of the safe range for a heart rate. The dashed line represents the average heart rate over the whole simulation time. In this graph, each point represents the value of a heart rate over one time window, the bottom axis represents the number of time windows over the simulation time. In Fig. 4 (right), we show an example run where tachycardia is exhibited even if the ranges of the given input corresponds to a resting state. Table 1 summarizes the results for out heart rate anomaly detection. In about 9.2% of our experiments an error occurred. All the errors were related to a heart rate that is too high. The category Mean Heart Rate Tachycardia refers to when the average heart rate of an experiment was deemed too high and the temporary tachycardia refers to the case where only some heart rates were too high.

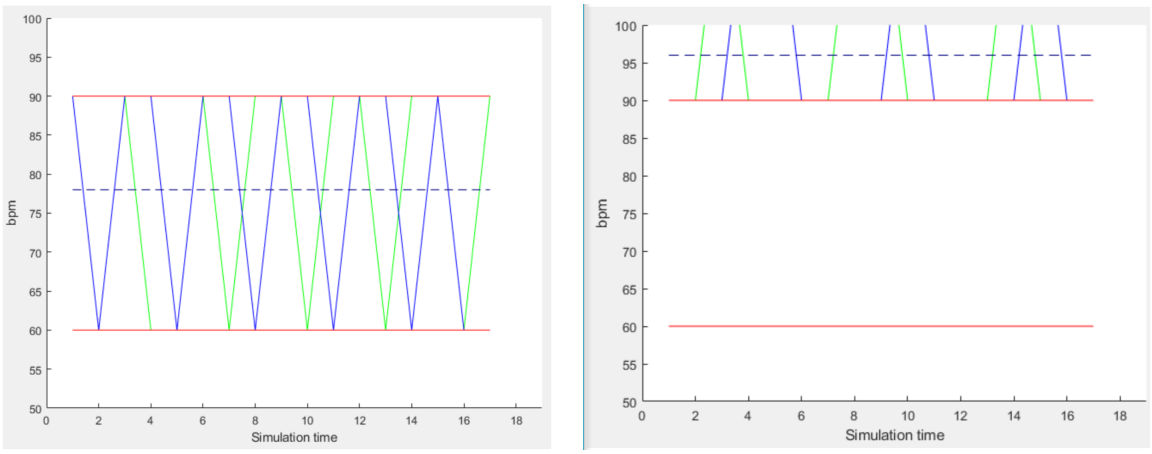


Fig. 4. Normal heart rate over time (left) and Tachycardia Behavior (right)

# Experiments	500
Mean Heart Rate Tachycardia	29
Temporary Tachycardia	17
Mean Heart Rate Bradycardia	0
Temporary Bradycardia	0
Total bugs found	46
Percentage of faulty runs	9.2%

Table 1. Results Summary for heart rate detection.

Rhythm Hijacking In order to detect rhythm hijacking, a condition where the pacemaker "hijacks" a normal pulsing and disallows the heart to depolarize by itself, we specifically looked at the atrioventricular signals (VA) and atrioventricular paces signals (VP) and calculated how many consecutive (VP) signals occurred in a row. To fuzz the model, we tested three scenarios: 1) randomization based off of normal parameter ranges, 2) randomization based on ranges stretched 20% in both direction, and 3) randomization based on ranges stretched 100% in both directions. For example, if a normal ERP period is 280 to 340 ms, a plus 20% range would result in a range of 278 ms to 352 ms.

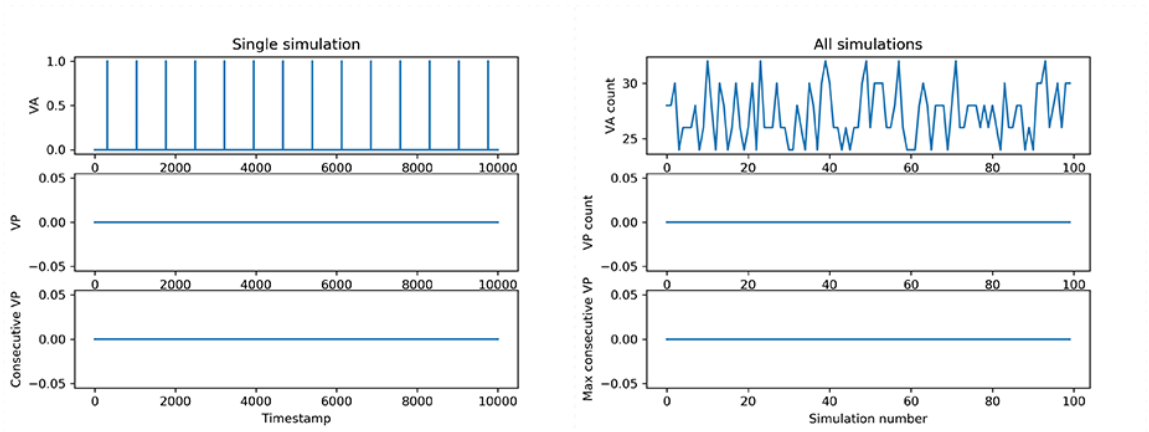


Fig. 5. Fuzzing with normal parameter ranges

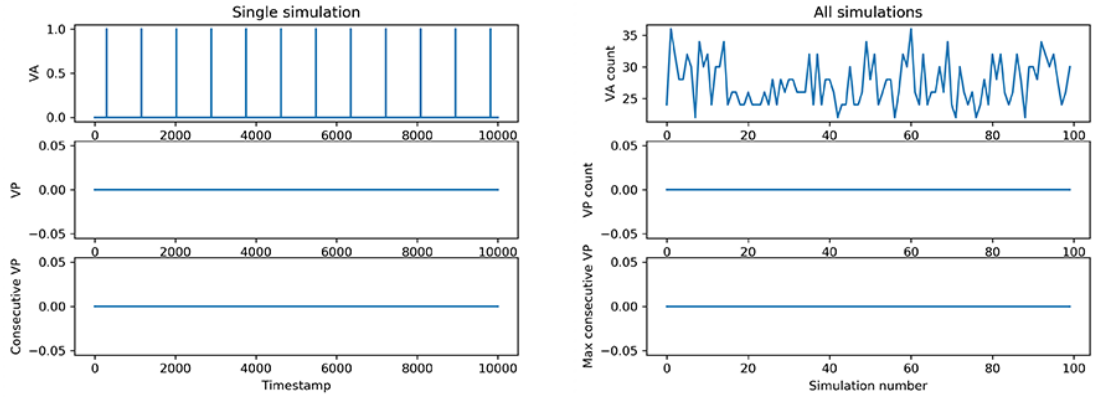


Fig. 6. Fuzzing with 20 % stretched parameter ranges

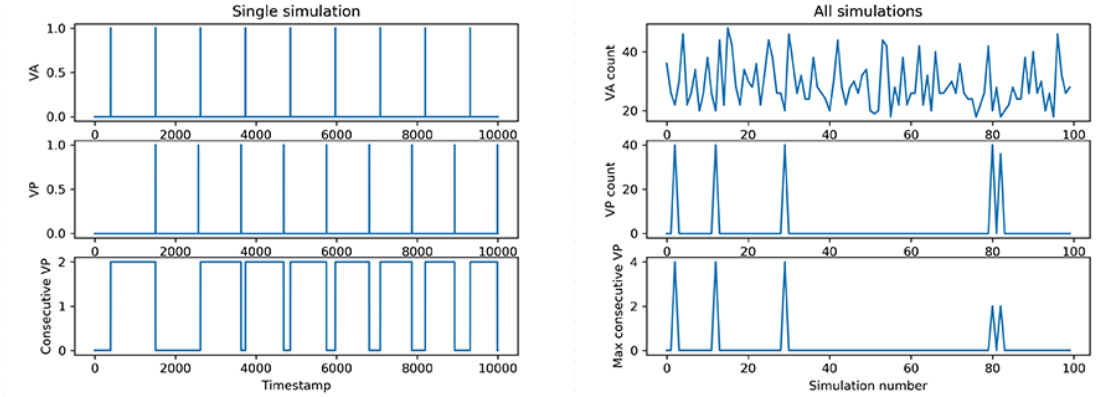


Fig. 7. Fuzzing with 100 % stretched parameter ranges

We did not detect any abnormal behavior, or any pacing behavior when we used normal ranges, or even ranges stretched by 20% as seen in Figure 5 and Figure 6. However, when we stretched the normal range by 100%, we encounter 5 instances of rhythm hijack over 100 runs as seen in Figure 7. While we counted the instances of detected rhythm hijacks as bugs, whether or not the detection of this condition within the specified range constitutes a faulty behavior depends on specifications of the manufacturers, and the historical upper and lower bound ranges that deal with the physical properties of human physiology.

A-to-V Delay The final property we look at is the delay between when a pulse is generated in the SA node and when it is felt in the AV node. Too long of a delay indicates that the pacemaker is working incorrectly because the pacemaker should induce an artificial heartbeat within a reasonable time frame. In algorithm 1, lines 19 to 24 are used to detect when the A-to-V signal is delayed, or if it occurs too soon.

Across 500 experiments, we recorded 55 cases where the conduction delay was higher than the defined upper bound of 200 ms. Despite this, the cases where the model violated the safety property had recorded conduction delays of 201 ms, indicating that the delay was barely higher than the threshold. None of the 500 experiments had a conduction delay lower than the defined lower bound of 50 ms. Figure 8 shows the recorded A-to-V delay violations for a run of 10 experiments.

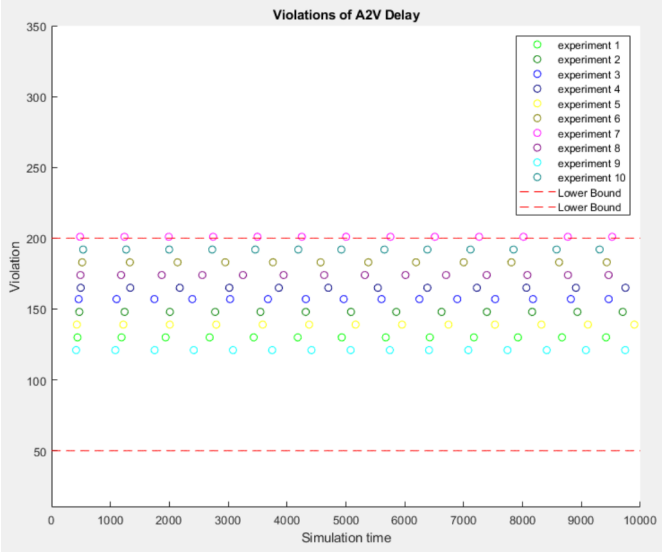


Fig. 8. A-to-V delay violations

5 FUZZING HR MODEL 2

5.1 Model Description

The second model is a Heart Systemic Pulmonary (HSP) model that models the human cardiovascular system, including the pulmonary and systemic circulatory systems, developed by Zhe Hu [4]. The model is a logic-based Simulink block model and is very complex,

as displayed in Fig. 9. The model is extremely detailed and models different aspects of the heart and circulatory systems functioning, including blood volume, pressure systems, blood pathway resistance through different system components, as well as the interactions and relationships between all of the circulatory systems and heart components (such as the ventricles.) The model uses 39 data inputs (parameters of real circulatory systems) and 47 control inputs (model constants). A picture showing the different data inputs and outputs with their system aspects and the component they impact is shown in Fig. 10. In order to fuzz this model, we will manipulate the data input parameters.

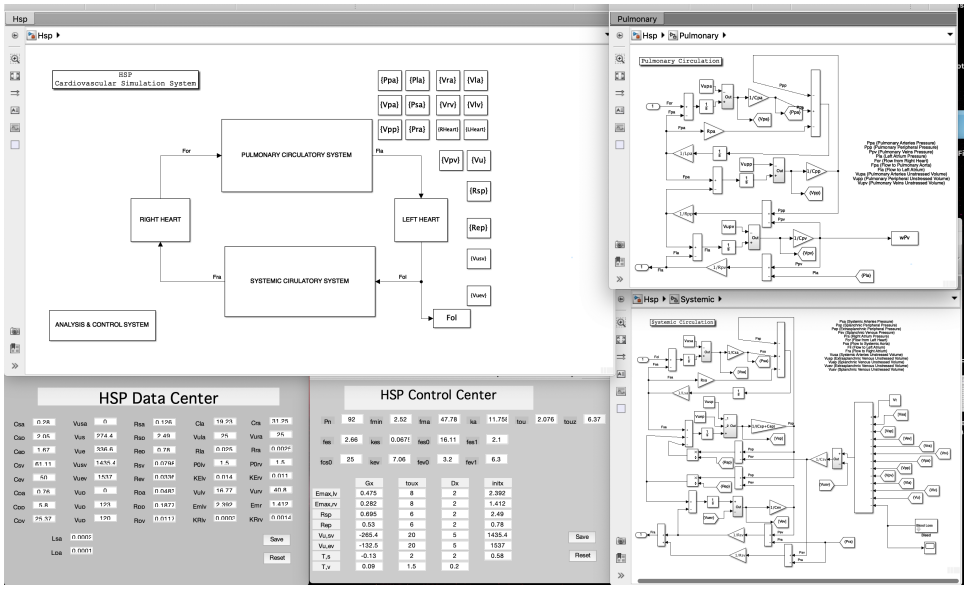


Fig. 9. Screenshot of the HSP block model and inner logic of the Pulmonary Circulatory and Systemic Circulatory systems.

Inputs. We elucidate the different data inputs used to fuzz our model. The model has general input parameters that provide the basis and assumption for the other parameters of a resting Heart Rate (HR) of 72 bpm and a 150 lb person, who has about 5,300 mL volume of blood. It also includes Inertance parameters that model pressure differences between different circulatory systems, Compliance parameters to model the amount of stressed blood volume stored at a given pressure, the Hydraulic Resistance (pressure energy losses in a specific heart compartment), the Unstressed Blood Volume modeling the amount of *unstressed* blood volume stored at a given pressure, Heart Beat parameters related to

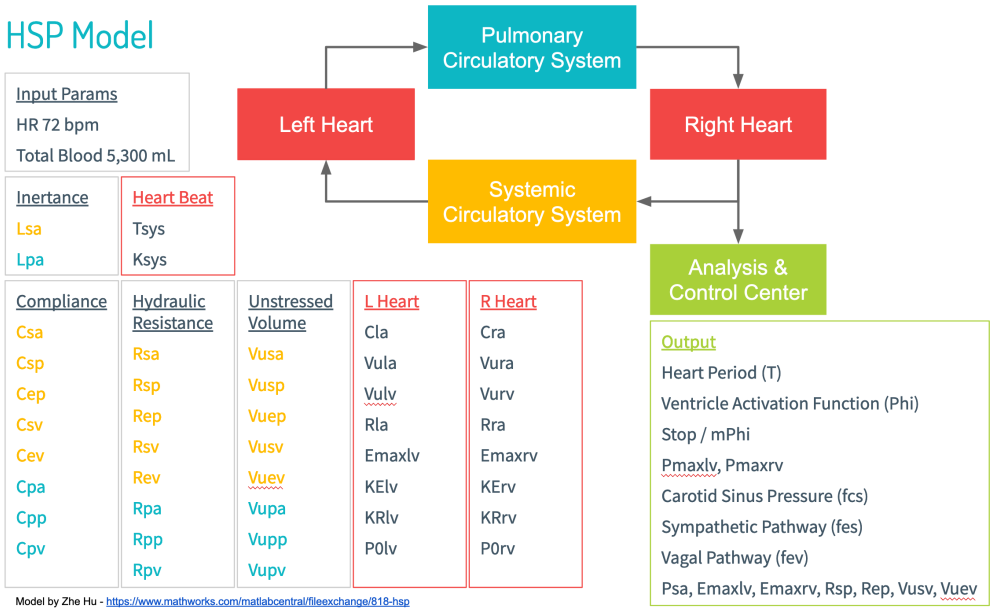


Fig. 10. HSP Model with Input Parameters.

systole slope and functioning, and parameters for the functioning of the Left and Right Heart. The input parameters are explained in greater detail in Table 2.

Outputs. Our model produces 15 outputs from the Analysis and Control Center of the HSP model. The outputs include the Heart Period (T); the Ventricle Activation Function (Phi); the function end point (stop/mPhi); Pmaxlv, Pmaxrv; Carotid Sinus Pressure (fcs); Sympathetic Pathway (fes); Vagal Pathway (fev); and a variety of factors to capture functioning of the different system components including Psa, Emaxlv, Emaxrv, Rsp, Rep, Vusv, and Vuev. The output parameters are explained in greater detail in Table 3.

5.2 Fuzzing Overview

Initially, we used random number generators to randomly generate different sets of data and control inputs. However, the model would break and not run the system because the data inputs were not in the correct medical ranges, and we did not realize that the control inputs are constants that do not change and therefore should not be fuzzed. As a result, we then had to develop an intelligent fuzzer that generated random input sets only for

Table 2. HSP Input Parameter Explanations

Param	Default Value	Subcategory	Heart Component
HR	72 bpm	General Input	Entire System
Total Blood	5300 mL	General Input	Entire System
Lsa	0.00022	Inertance	Systemic Circulatory
Lpa	0.00018	Inertance	Pulmonary Circulatory
Csa	0.28	Compliance	Systemic Circulatory
Csp	2.05	Compliance	Systemic Circulatory
Cep	1.67	Compliance	Systemic Circulatory
Csv	61.11	Compliance	Systemic Circulatory
Cev	50	Compliance	Systemic Circulatory
Cpa	0.76	Compliance	Pulmonary Circulatory
Cpp	5.8	Compliance	Pulmonary Circulatory
Cpv	25.37	Compliance	Pulmonary Circulatory
Rsa	0.06	Hydraulic Resistance	Systemic Circulatory
Rsp	3.307	Hydraulic Resistance	Systemic Circulatory
Rep	1.407	Hydraulic Resistance	Systemic Circulatory
Rsv	0.038	Hydraulic Resistance	Systemic Circulatory
Rev	0.016	Hydraulic Resistance	Systemic Circulatory
Rpa	0.023	Hydraulic Resistance	Pulmonary Circulatory
Rpp	0.0894	Hydraulic Resistance	Pulmonary Circulatory
Rpv	0.0056	Hydraulic Resistance	Pulmonary Circulatory
Vusa	0	Unstressed Blood Vol	Systemic Circulatory
Vusp	274.4	Unstressed Blood Vol	Systemic Circulatory
Vuep	336.6	Unstressed Blood Vol	Systemic Circulatory
Vusv	1121	Unstressed Blood Vol	Systemic Circulatory
Vuev	1375	Unstressed Blood Vol	Systemic Circulatory
Vupa	0	Unstressed Blood Vol	Pulmonary Circulatory
Vupp	123	Unstressed Blood Vol	Pulmonary Circulatory
Vupv	120	Unstressed Blood Vol	Pulmonary Circulatory
Cla	19.23	Heart	Left Heart
Vula	25	Heart	Left Heart
Vulv	16.77	Heart	Left Heart
Rla	0.00025	Heart	Left Heart
Emaxlv	2.95	Heart	Left Heart
KElv	0.014	Heart	Left Heart
KRlv	0.000375	Heart	Left Heart
P0lv	1.5	Heart	Left Heart
Cra	31.25	Heart	Right Heart
Vura	25	Heart	Right Heart
Vurv	40.8	Heart	Right Heart
Rra	0.00025	Heart	Right Heart
Emaxrv	1.75	Heart	Right Heart
KErv	0.011	Heart	Right Heart
KRrv	0.00014	Heart	Right Heart
P0rv	1.5	Heart	Right Heart

Table 3. HSP Output Parameter Explanations

Param	Meaning	Affected Component
T	Heart Period	Heart
Phi	Ventricle Activation Function	Heart Ventricles
Stop / mPhi	Activation Function Endpoint	Heart Ventricles
Pmaxlv	Ventricle Pressure	Left Heart
Pmaxrv	Ventricle Pressure	Right Heart
Fcs	Carotid Sinus Pressure	Systemic and Pulmonary Circulatory
Fes	Sympathetic Pathway	Systemic and Pulmonary Circulatory
Fev	Vagal Pathway	Systemic and Pulmonary Circulatory
Psa	Blood Pressure	Heart
Emaxlv	Systole and Slope Relationship	Heart
Emaxrv	Systole and Slope Relationship	Heart
Rsp	Hydraulic Resistance Factor	Systemic Circulatory
Rep	Hydraulic Resistance Factor	Systemic Circulatory
Vusv	Unstressed Vol Factor	Systemic Circulatory
Vuev	Unstressed Vol Factor	Systemic Circulatory

the data parameters, using the known medical ranges and then develop the detector to identify expected vs anomalous outputs.

In order to fuzz our model, we tested two different aspects: first, we fuzzed the system using random input parameters (within normal ranges and represented as signals over time) and check for violations of known medical properties. Second, we adapted the model for different individuals (assuming different resting heart rates and blood volumes) and adjusted the input parameters accordingly to see what bugs may be found this way. Following the overview given of the fuzzer in Section 3, we developed a simulation script to generate random inputs based on medical knowledge and to send the inputs to the Simulink model and run the model. We also developed detector scripts to determine when the output was correct or not. These will be explained in greater detail in the succeeding section. The HSP Fuzzer pseudo code is shown in Algorithm 2.

Algorithm 2 Hsp-Fuzzer(nExperiments, scaleFactor, errorThreshold, contractilityProp, resistanceProp, unstressedProp)

****** Set Fuzzing Params ******
****** Generate Random Fuzzer Inputs ******
for $i = 1$ to $nExperiments$ **do**
    for all ip in inputParams, $ip[i] = ip[i] * scaleFactor$
    if $contractilityProp = True$ **then**
        randNum = random number(1-33%)
        for all c in contractilityParams $c[i] = c[i] * randNum$
    end if
    if $resistanceProp = True$ **then**
        randNum = random number(200-210%)
        for all r in resistanceParams $r[i] = r[i] * randNum$
    end if
end for
****** Start Detector ******
$origOutput = loadorig.mat$ ▷ Load original output signals for comparison
out = get 15 output vars from model
for all op in outputParams, origOp in origOutput **do**
    if abs[op - (origOp * scaleFactor)] > errorThreshold **then**
        scaleBugs++ ▷ Detect Scale Factor Bugs
    end if
    if abs(op.fcs - origOp.fcs) < 50 || abs(op.fcs - origOp.fcs) > 200 **then**
        contractilityBugs++ ▷ Detect Contractility Bugs
    end if
    if origOp.fcs - op.fcs > 201 || origOp.fcs - op.fcs < 0 **then**
        resistanceBugs++ ▷ Detect Resistance Bugs
    end if
end for
for b=2 to length(op.vusv) **do**
    if abs[op.vusv(b-1) - op.vusv(b)] > 15 **then**
        unstressedBugs++ ▷ Detect Unstressed Bugs
    end if
end for
Print scaleBugs, contractilityBugs, resistanceBugs, unstressedBugs

5.3 Fuzzing Medical Properties

Description. We checked 3 medical properties when randomly fuzzing the input parameters, as described in [6]. These medical properties can be understood basically as the specific relationships between inputs and outputs. The first property is Contractility. This property states that systole slope (Emaxrv and Emaxlv) can change by up to 33%, and the expected output is that the Cartoid Sinus Pressure (Fcs) will change by 50-200mm/Hg. The second property is Resistance. This property states that the Hydraulic Resistance parameters can increase by 200-210%, and the expected output is that the Cartoid Sinus Pressure (Fcs) will decrease by 50-200 mm/Hg. The third and final property we checked was Unstressed Blood Volume. This property states that the change in Unstressed Blood Volume from timestep to timestep should always be less than 16 mL/kg. These properties were chosen because they are three of the main ones the HSP model authors argue they support [4]. As such, the models should be able to hand fuzzed inputs that affect these properties. However, for future other properties may be incorporated and checked. Our detector script for these properties relies on knowing the expected, normal output. *A change in a parameter input that does not produce the expected output is a **bug**.* The random inputs are generated for the properties in Algorithm 2 Lines 5-12 and detected using our detector in Lines 21-31.

Results. To test our fuzzer and the HSP model, we fuzzed random input parameters for 100 iterations, with each input parameter having about 15,000 time steps. In terms of metrics to evaluate our fuzzer, we used coverage and number of bugs found. We looked at edge and block coverage (at a literal level since the Simulink model consists of block components and edge relationships between components,) and were able to cover 100% of the model. This is because all of our inputs touch and affect every single block and edge of the model. As a result, when we fuzz all of these different inputs, we achieve 100% coverage. An example of the coverage for the Pulmonary Circulatory System is shown in Fig 11. We found many bugs for the first two properties, but none for the third property, as shown in Table 4.

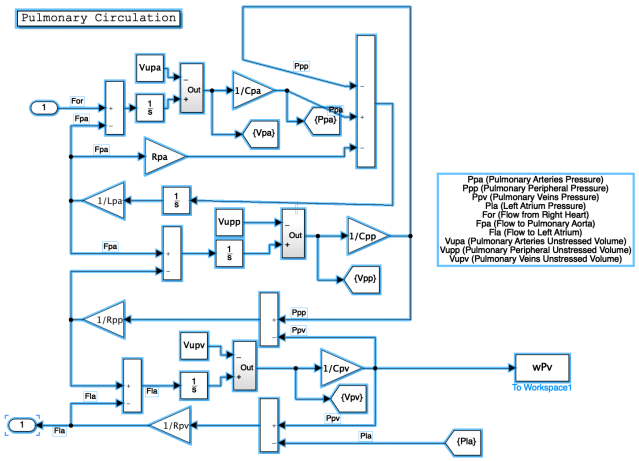


Fig. 11. Coverage of Fuzzer for Pulmonary Circulatory System. Fuzzer achieves 100% coverage.

Table 4. HSP Output Parameter Explanations

Property	Total Bugs Found	Average Bugs Per Iteration
1 (Contractility)	1,320,912	13,209
2 (Resistance)	361,039	3,610
3 (Unstressed Vol)	0	0

5.4 Fuzzing by Scale Factor

Description. The original HSP model assumes set general input parameters for the resting heart rate and blood volume, used to derive all of the other input parameters for different parts of the system. It uses a 72 bpm resting heart rate and a 150 lb person who has about a 5,300 mL blood volume. However, this assumption does not hold for every human, and as a result, we needed to change our input assumptions and fuzz for different input parameters. We changed the resting heart rate and blood volume, and used the generated heart rates from Model 1. Our input consisted of increasing or decreasing the parameters by a scale factor to represent different people. For instance, if we had a 300 lb person, this means their blood volume would be about double a 150 lb person (so 5,300 x 2, indicating a scale factor of 2.) When we scale the input parameters, the specific relationships between the data variables should not change. As such, our output is that the relationships between the parameters should not change by more than x units, (some predefined error threshold.)

*Any change more than this error threshold represents a **bug**.* The scale factor inputs are generated in Algorithm 2 Line4, and detected using our detector in Lines 18-20.

Results. Overall, we found many bugs occurred as we changed the scale factor, indicating the model does not properly handle input parameters representing diverse humans. We first looked at the number of bugs found by output variable, as shown in Fig. 12. As we can see, many of the system-wide parameters such as the Stop and T parameters do not have any bugs, whereas the very specific heart and circulatory system parameters such as Fcs, Fes, Pmaxlv, Pmaxrv, Vuev, and Vusv have many bugs found.

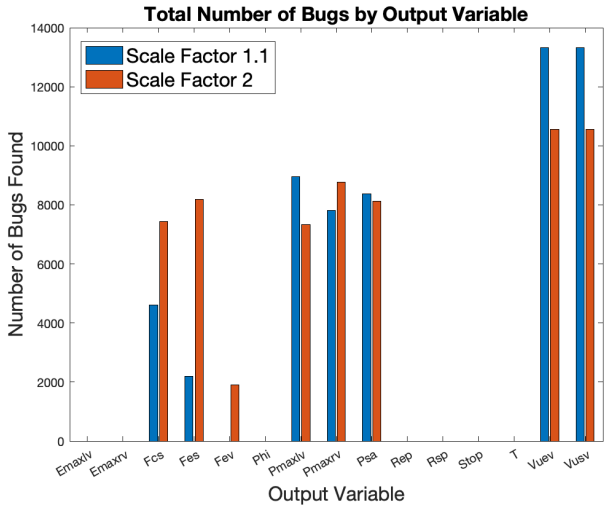


Fig. 12. Bugs found by output variable for Scale Factors of 1.1 and 2.

Next, we looked at the relationship between the scale factor used and the number of bugs found, as shown in Fig. 13. A scale factor of 1 indicates we did not change the scale (which is why there are 0 bugs on the graph at this point.) As shown, the number of bugs found changes depending on what the scale factor is. This is an interesting finding, as it indicates that the model may be more equipped to handle certain types of individuals (i.e. those that weigh less and have less blood volume,) compared to others (i.e. those that weigh more and have more blood volume.) This information could be used to adapt and improve the model.

We also looked at the relationship between the error threshold and the number of bugs found as shown in Fig 14. Since there was no medical literature specifically related to

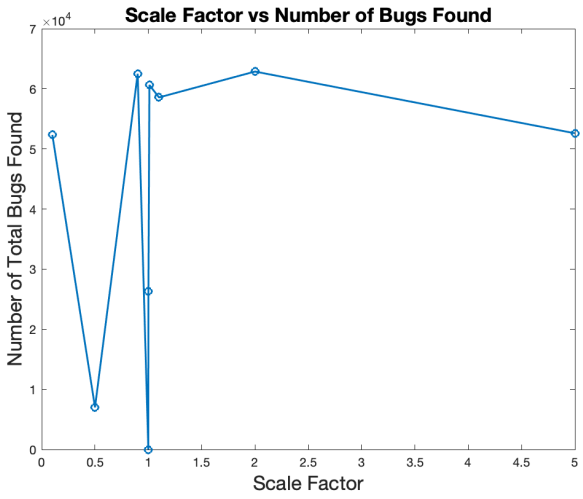


Fig. 13. Scale Factors vs Bugs Found

finding a good error threshold, we wanted to explore the range of bugs found based on changing the threshold. As can be seen from the graph, as we make the error threshold more stringent (i.e. only 1 or 2 units of error allowed,) the number of bugs increases, and as we relax the error threshold (allow more error), the number of bugs decreases. This makes sense and is a logical finding. For our other experiments, we decided a threshold of 5 units seemed like a good value.

Finally, we were also able to visualize the actual input signals themselves in order to see how the fuzzer affected their output. In Fig. 15, we can see the difference between the normal Carotid Sinus Pressure signal and the anomalous signal from the fuzzed input, using a scale factor of 2. The output signal goes to near 0, which is physically impossible in the human body, clearly showcasing the identified bug. Another example for the left and right ventricles is shown in Fig. 16. The fuzzed ventricle signals look very different from the normal ones, indicating an identified bug for the two signals.

6 CONCLUSION

One of the main learning experiences of this project for our group is that fuzzing Cyber-physical systems in simulation is significantly different from fuzzing executables in that

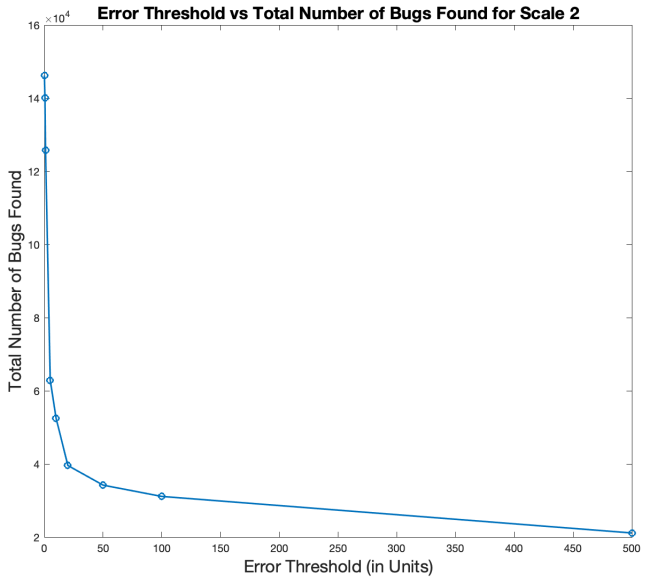
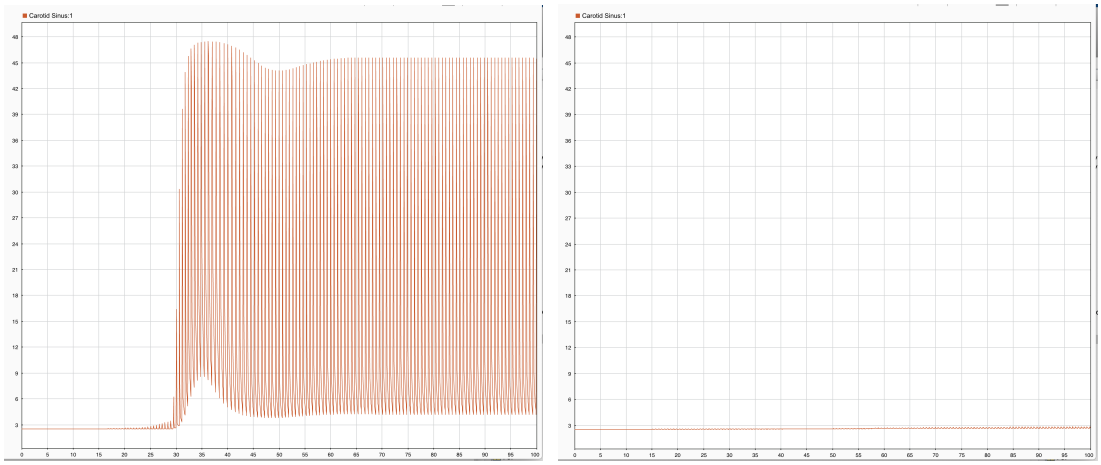
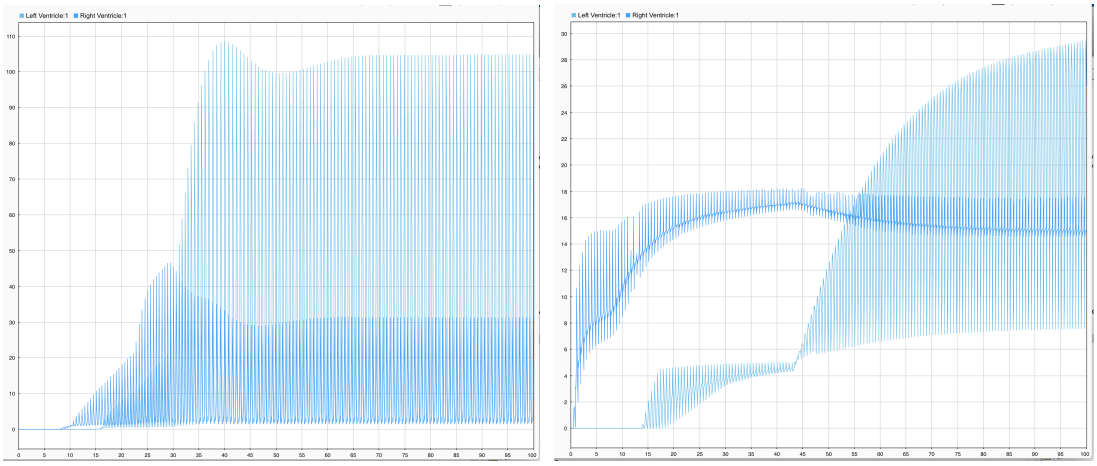


Fig. 14. Error Threshold vs Bugs found



(a) (b)

Fig. 15. **(a)** Normal and **(b)** Fuzzed Carotid Sinus Pressure Signals assuming a scale factor of 2 and an error threshold of 5.



(a) (b)

Fig. 16. **(a)** Normal and **(b)** Fuzzed Left and Right Ventricle Signals assuming a scale factor of 2 and an error threshold of 5.

the failure condition is not the program crashing, but rather the resultant behavior being outside the range of prescribed behavioral boundaries. This presents an important challenge when applying the concept of fuzzing to any domain specific applications, as each failure condition also requires specifically handcrafted detection which is tailored to the domain and model. In addition to concerns regarding the scalability of fault detection on the output, the input parameter randomizations are also subject to the same domain specific restrictions. Not bounding the randomization range to sensible values (i.e. having a heart rest period of -5 milliseconds) raises questions regarding relevance and meaningful physical representation. In spite of all these constraints, however, our naive value randomization schemes through input range scaling were able to reveal failure conditions which would be detrimental in a real-life scenario. As such, while many restrictions apply, the value for fuzzing cyber-physical systems in a world of increasing complexity cannot be understated.

REFERENCES

[1] ID: 50417. 2019. DeepFuzzer. https://github.com/50417/DeepFuzzer.

[2] Shafiul Azam Chowdhury, Taylor T Johnson, and Christoph Csallner. 2016. CyFuzz: A differential testing framework for cyber-physical systems development environments. In *International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems*. Springer, 46–60.

[3] Shafiul Azam Chowdhury, Soumik Mohian, Sidharth Mehra, Siddhant Gawsane, Taylor T Johnson, and Christoph Csallner. 2018. Automatically finding bugs in a commercial cyber-physical system development tool chain with SLforge. In *Proceedings of the 40th International Conference on Software Engineering*. ACM, 981–992.

[4] Zhe Hu and Ying-min Diao. 2002. Primary model of heart-systemic-pulmonary system. *JOURNAL-TONGJI UNIVERSITY* 30, 1 (2002), 61–65.

[5] 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.

[6] Mauro Ursino. 1998. Interaction between carotid baroregulation and the pulsating heart: a mathematical model. *American Journal of Physiology-Heart and Circulatory Physiology* 275, 5 (1998), H1733–H1747.